Modelagem e Validação de um Sistema de Determinação de Atitude com Tolerância a Falhas para o NanosatC-Br2

Bruno Caetano de Oliveira Miranda

UFMG

Universidade Federal de Minas Gerais
Escola de Engenharia
Programa de Pós-Graduação em Engenharia Elétrica

Belo Horizonte
2016

Bruno Caetano de Oliveira Miranda

Modelagem e Validação de um Sistema de Determinação de Atitude com Tolerância a Falhas para o NanosatC-Br2

Dissertação de mestrado submetida ao Curso de Pós-graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Mestre em Engenharia Elétrica.

Área de concentração: Sistemas de Computação e Telecomunicações
Linha de Pesquisa: Microeletrônica e Microssistemas (MeMSs)

Orientador: Prof. Dr. Ricardo de Oliveira Duarte

Belo Horizonte
2016

Este trabalho é dedicado a todos aqueles que destinam uma pequena fração de sua existência a essa leitura.

Agradecimentos

Ao professor Dr. Ricardo de Oliveira Duarte, por todas as orientações e discussões (técnicas ou não) que foram fundamentais ao desenvolvimento desse trabalho; e por me apresentar o projeto do SDATF, que exerce importante função no Programa Espacial Brasileiro.

À minha família, em especial aos meus pais, que enxergaram a importância da educação e da instrução e souberam dar a devida prioridade a tais benéfices, o que fez e continua fazendo toda a diferença em minha vida. À minha namorada, parte importante da minha vida, por toda a paciência perante as dificuldades e por dar sentido ao que faço.

A todos do Centro de Engenharia e Tecnologia da Embraer Minas Gerais, pelo apoio a essa empreitada.

Ao meu país, de onde obtive toda instrução e em relação ao qual possuo eterna dívida.

À força de inteligência e bondade superior que permeia cada fração desse universo, seja qual for sua natureza.

Ao senhor John Williams, por compor as mais belas trilhas sonoras conhecidas por essa civilização, as quais foram intensamente ouvidas ao longo do desenvolvimento do trabalho aqui apresentado.

Finalmente, a todos que contribuíram direta ou indiretamente para que esse trabalho fosse realizado.

"In a dark place we find ourselves, and a little more knowledge lights our way. (...) Luminous beings are we. Not this crude matter."

"Em um lugar escuro nos encontramos, e um pouco mais de conhecimento ilumina nosso caminho. (...) Seres luminosos somos nós. Não esta matéria bruta."

Mestre Yoda - Star Wars, Episódios III e V

Resumo

O trabalho em questão consiste na validação por modelagem e simulação do Sistema de Determinação de Atitude com Tolerância a Falhas (SDATF), a ser testado na baixa órbita terrestre como carga útil do satélite científico NanosatC-Br2. Tal satélite vem sendo desenvolvido pela equipe do INPE (Instituto Nacional de Pesquisas Espaciais) em conjunto com a UFSM (Universidade Federal de Santa Maria) e tem por objetivo, além de consolidação da tecnologia associada no Brasil, o envio de experimentos científicos ao espaço. Devido à grande exposição de circuitos à radiação em órbita, um sistema de determinação de atitude usado em satélite deve ser robusto a chaveamento de bits decorrentes dessa exposição, fenômeno denominado *Single Event Effect* (SEE). Quando um SEE atinge uma célula de memória, corrompendo a informação armazenada, tem-se um *Single Event Upset* (SEU). O SDATF, portanto, se propõe a fornecer informações de atitude ao computador de bordo do satélite por meio de uma solução tolerante a falhas do tipo SEU. Esforços preliminares já foram feitos com o intuito de se validar o SDATF, mas devido a possíveis melhorias e alterações de requisitos ao longo do projeto, um método de validação mais flexível e que reduza o retrabalho se faz necessário. Embora a validação por modelagem e simulação vá de encontro a essa necessidade, deve-se estabelecer um procedimento sistemático para a criação de um modelo fiel ao comportamento real do sistema, visto que características como escalonamento de tarefas, paralelismo e não determinismo tornam a descrição formal do sistema de maneira não ambígua uma tarefa complexa. O trabalho que se segue apresenta um procedimento para modelagem e simulação de sistemas com tolerância a falhas do tipo SEU, instanciando esse método ao SDATF. Ao longo do trabalho são abordados conceitos relacionados aos efeitos de partículas de alta energia sobre circuitos integrados no ambiente espacial, as causas desses efeitos e formas de mitigação. O formalismo necessário à modelagem do sistema é apresentado, bem como os experimentos com injeção simulada de falhas e os resultados que reforçam a validade do sistema proposto.

Palavras-chave: Single Event Upset; Tolerância a Falhas; Validação; Modelagem.

Abstract

This work consists in validation by model and simulation of the Fault Tolerant Attitude Determination System (SDATF), to be tested in terrestrial low orbit as payload of the scientific satellite NanosatC-Br2. This satellite is being developed by the National Institute for Space Research (INPE) in collaboration with Santa Maria Federal University, and its objectives are, besides the associated technology consolidation in Brazil, sending scientific experiments to space environment. Due to large exposure of the circuits to radiation in orbit, the attitude determination system used in a satellite shall be robust to bit flips caused by this exposure, phenomena denoted as *Single Event Effect* (SEE). When a SEE affects a memory cell, corrupting the stored information, it is said that a *Single Event Upset* (SEU) occurred. SDATF, therefore, is proposed to provide information about the attitude to satellite's onboard computer by means of a SEU fault tolerant solution. Preliminary efforts were already made with the purpose of validating SDATF, but a more flexible and rework reducing validation method is necessary due to possible system improvements and requirements changes along project lifecycle. Although validation by model and simulation meets this necessity, a sistematic procedure to build a realistic model shall be established, since features like task scheduling, parallel operation and non determinism make an unambiguous formal description of the system a complex task. The work that follows presents a procedure for modelling and simulation of SEU like fault tolerant systems, instantiating this method with SDATF. Along this work, concepts related to the effects of high energy particles on integrated circuits in the space environment, their causes and ways of mitigation are covered. The necessary formalism for modelling the system is presented, as well as the experiments with simulated fault injection and the results that endorse the validity of the system proposed.

Keywords: Single Event Upset; Fault Tolerance; Validation; Modelling.

Lista de ilustrações

Lista de tabelas

Lista de siglas

ACK	*Acknowledgement*
BJT	*Bipolar Junction Transistor*
CEU	*Coding Emulating Upset*
CI	Circuito Integrado
CMOS	*Complementary Metal Oxide Semiconductor*
COTS	*Commercial-Off-The-Shelf*
FET	*Field-Effect Transistor*
FPGA	*Field Programmable Gate Array*
FSM	*Finite State Machine*
HEXTE	*High Energy X-Ray Timing Experience*
HSO	*Heliophysics System Observatory*
INPE	Instituto Nacional de Pesquisas Espaciais
LABSIM	Laboratório de Simulação Instituto Nacional de Pesquisas Espaciais
MC	Microcontrolador
MOSFET	*Metal Oxide Semiconductor Field-Effect Transistor*
MTTF	*Mean Time To Failure*
MTTR	*Mean Time To Repair*
NASA	*National Aeronautics and Space Administration*
OBC	*OnBoard Computer*
PSD	*Position Sensitive Detector*
QUEST	*Quaternion Estimator*
RXTE	*Rossi X-Ray Timing Explorer*
SAA	*South Atlantic Anomaly*
SDATF	Sistema de Determinação de Atitude com Tolerância a Falhas
SEE	*Single Event Effect*
SET	*Single Event Transient*
SEU	*Single Event Upset*

UART *Universal Asynchronous Receiver/Transmitter*
UFRGS Universidade Federal do Rio Grande do Sul
UFSM Universidade Federal de Santa Maria

Sumário

Capítulo 1

Introdução

1.1 Contexto e Motivação

Nos últimos anos a utilização dos chamados nanossatélites (satélites com peso inferior a 1 kg) em aplicações científicas vem crescendo devido ao seu baixo custo se comparados aos satélites convencionais. Em particular, os chamados cubesats - nanossatélites em formato cúbico, com aresta de aproximadamente 10 cm - tem se mostrado soluções interessantes no envio de experimentos científicos à baixa órbita terrestre. Os cubesats tiveram origem em 1999, por meio de esforços de grupos da California Polytechnic State University (*Cal Poly*) e da Universidade de Stanford (HELVAJIAN; JANSON, 2008). A Figura 1 mostra uma representação de um cubesat.

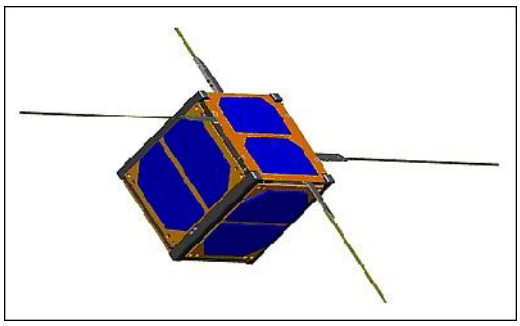


Figura 1 – Modelo tridimensional de um cubesat.

Fonte: CRS/CCR/INPE-MCT

Em junho de 2014, o Instituto Nacional de Pesquisas Espaciais (INPE) em colaboração com o Laboratório de Ciências Espaciais de Santa Maria tiveram o primeiro cubesat brasileiro lançado à baixa órbita, o NanosatC-Br1 (SCHUCH; DURÃO,).

Atualmente, uma nova versão do NanosatC-Br está em desenvolvimento, o NanosatC-Br2. O objetivo desses satélites, além da consolidação da tecnologia associada no Brasil, é a realização de experimentos em baixa órbita.

Um dos experimentos a ser embarcado no NanosatC-Br2 é o Sistema de Determinação de Atitude com Tolerância a Falhas, ou SDATF (DUARTE et al., 2008). Trata-se de um circuito voltado para a determinação da orientação do satélite por meio de um algoritmo que leva em conta medições do campo magnético e da luz solar.

No entanto, a exposição dos circuitos de instrumentação à radiação no ambiente espacial constitui um problema compartilhado entre as diversas classes de satélites. Subsistemas como o de determinação de atitude são baseados em sensores e microcontroladores, os quais estão sujeitos a fenômenos conhecidos como *Single Event Effects* (SEE). Os SEEs são causados pelo choque de partículas de alta energia, tais como prótons, nêutrons e íons em regiões sensíveis do material semicondutor dos dispositivos integrados (DODD; MASSENGILL, 2003; TORRES, 2013). Essas colisões podem gerar alterações de tensão e níveis lógicos de caráter temporário ou permanente em regiões de memória, ocasionando o mal funcionamento do circuito. Um tipo específico e mais frequente de SEE é o chamado *Single Event Upset* (SEU), que consiste na alteração do nível lógico da uma unidade de memória sem causar danos permanentes ao circuito.

O SDATF é uma proposta de solução tolerante a falhas, baseado em redundância de microcontroladores COTS (*Commercial-Off-The-Shelf*), cujo objetivo é prover informação a respeito da atitude de forma consolidada e em alta disponibilidade, mesmo em um ambiente sujeito à ocorrência de SEUs.

As soluções de circuitos tolerantes a falha podem ser baseadas em hardware (KEYMEULEN et al., 2000) ou software (GARG, 2010). Soluções baseadas em hardware podem envolver encapsulamento especial ou reconfiguração dinâmica do circuito (KEYMEULEN et al., 2000), o que pode elevar o custo do projeto. Já as soluções baseadas em software levam em conta a redundância de elementos de hardware de uso geral, o que muitas vezes resulta em um custo menor. Nesse caso, as soluções se diferenciam pelo algoritmo utilizado para sincronizar e integrar os elementos redundantes (DUARTE et al., 2008; GARG, 2010).

Apesar de permitir o projeto de subsistemas espaciais a um baixo custo, soluções com tolerância a falhas aumentam a complexidade do sistema, dado que novas características devem ser consideradas, tais como o sincronismo dos componentes, a alta disponibilidade, o não determinismo e o paralelismo. De acordo com Lee e Seshia (LEE; SESHIA, 2011), o desenvolvimento recente de sistemas embutidos vem encontrado maior desafio na interação com o ambiente físico, o que se aplica ao problema de ocorrência de falhas no ambiente espacial. Dessa forma, soluções tolerantes a falhas devido a partículas de alta energia envolvem certa complexidade em seu processo de validação, uma vez que as condições do espaço não são facilmente reproduzidas na Terra.

Esforços preliminares já foram feitos com o intuito de se validar o SDATF (TORRES, 2013; ROSA, 2014). Tais medidas envolveram a emulação de falhas via software na operação do circuito real. Porém, ao longo do projeto pode haver alteração de requisitos,

bem como propostas de melhorias na solução, de forma que um processo de validação a nível do design se mostra mais adequado à solução do problema.

Uma nova proposta de validação, baseada em modelagem e simulação, será apresentada nesse trabalho. O desafio dessa proposta consiste em capturar as características importantes relacionadas à operação do sistema sem de fato implementá-lo na forma de protótipo. O intuito é substanciar a ideia de que a solução apresenta resultado satisfatório, de forma a flexibilizar o desenvolvimento e aumentar a maturidade nas fases iniciais do projeto.

Ao longo desse trabalho será apresentado um procedimento sistemático para modelagem e simulação de um sistema tolerante a falhas do tipo SEU. Esse processo será instanciado para validação do SDATF, embora a mesma proposta possa ser estendida a outras soluções de mesma natureza.

1.2 Objetivos do Trabalho

O trabalho que se segue tem como objetivo a descrição de um procedimento de validação de sistemas tolerantes a falha do tipo *Single Event Upset*, a qual será utilizada para validação do Sistema de Determinação de Atitude com Tolerância a Falhas para o NanosatC-Br2 (SDATF).

1.3 Justificativa e Relevância

O envio do NanosatC-Br à baixa órbita representou um importante passo para o Programa Espacial Brasileiro. Espera-se que o NanosatC-Br2 dê continuidade às atividades científicas nacionais na baixa orbita terrestre. A consolidação de tecnologias espaciais aplicáveis a cubesats contribuirá para o avanço de um setor que tende a crescer nos próximos anos, em continuidade ao que já vem ocorrendo.

Conforme será abordado nesse trabalho, os efeitos decorrentes da colisão de partículas de alta energia em circuitos integrados tendem a ser mais significativos à medida em que novas tecnologias de fabricação de circuitos são desenvolvidas, haja vista que a redução nas dimensões dos transistores continua aumentando com o passar dos anos. O aproveito dos avanços de design e tecnologias atuais existentes nos circuitos integrados COTS modernos e a mitigação de tais efeitos oriundos da radiação na construção de sistemas construídos a partir de tais componentes são de grande importância para o Programa Espacial Brasileiro, visto que o nosso país está localizado em uma região do globo onde esses fenômenos podem ocorrer com mais frequência, fato que será abordado na Seção 2.4.

De uma forma geral, o processo de validação de sistemas é uma importante etapa no desenvolvimento de sistemas suscetíveis à ocorrência de falhas, de forma que a metodo-

logia que aqui se apresenta pode ser aplicada a demais sistemas com tolerância a falhas. A abordagem por modelagem (validação a nível de sistema, em detrimento da validação a nível de hardware ou software) se justifica pela não necessidade de fabricação de protótipos, o que propicia um aumento na maturidade da solução a um baixo custo, e com maior flexibilidade.

Do ponto de vista tecnológico, esse trabalho se posiciona como uma etapa no desenvolvimento de um sistema que, uma vez comprovada sua eficácia, poderá integrar sistemas espaciais baseados em tecnologia nacional.

Do ponto de vista científico, uma contribuição é dada à área de sistemas com tolerância a falhas e à modelagem e simulação de sistemas complexos.

1.4 Organização do Texto

Uma vez introduzido o assunto central desse trabalho, a dissertação segue com outros 6 capítulos.

No Capítulo 2 é feita uma discussão a respeito dos efeitos das partículas de alta energia em circuitos integrados, formalizando conceitos aqui citados, como *Single Event Effects* e *Single Event Upsets*. A ideia é apresentar o tipo de falha com que o SDATF deve lidar em sua operação, caracterizando esses fenômenos no contexto da baixa órbita terrestre.

O Capítulo 3 é dedicado à descrição do sistema tolerante a falhas. Seu funcionamento, bem como sua estrutura, são abordados em detalhes e servem como referência ao processo de modelagem e validação.

Nos capítulos 4, 5 e 6 o procedimento para modelagem e validação do sistema tolerante a falhas é apresentado. No capítulo 4 aborda-se um formalismo adequado para modelar de forma simples e não ambígua o comportamento do sistema a ser validado. No capítulo 5 são discutidos os aspectos práticos da criação do modelo de simulação no ambiente Simulink. Já no capítulo 6 são descritos os experimentos envolvendo injeção de falhas no modelo criado e medição do desempenho do sistema.

Por fim, no capítulo 7 são traçadas conclusões obtidas a partir desse trabalho e são feitas as considerações finais.

Capítulo 2

Efeitos de Partículas de Alta Energia em Circuitos Integrados CMOS

Nas seções a seguir serão abordadas questões referentes à interação de partículas de alta energia em circuitos integrados. Pretende-se caracterizar e contextualizar as falhas do tipo *Single Event Upset* (SEU) para, nos capítulos seguintes, descrever o circuito a ser validado e o método de validação propriamente dito.

2.1 Single Event Effects e Single Event Upsets

Transistores FET (*Field-Effect Transistor*, ou Transistor de Efeito de Campo) geralmente são menores e menos sensíveis à variação da temperatura do que os transistores BJT (*Bipolar Junction Transistor*, ou Transistor de Junção Bipolar). Além disso, os transistores FET apresentam consumo de energia muito inferior aos transistores BJT (e, consequentemente, menor dissipação de calor), o que faz com que os primeiros sejam largamente utilizados em circuitos integrados (CI) (BOYLESTAD; NASHELSKY, 1984), tais como os sistemas microcontrolados.

Entre os transistores FET, os do tipo MOSFET (*Metal Oxide Semiconductor Field-Effect Transistor*) são os mais utilizados na construção de CI's devido à sua estabilidade térmica e baixo consumo de energia. Os circuitos compostos por transistores MOSFET cujos terminais de fonte e dreno são compostos por material com dopagem do tipo p (pMOS) e do tipo n (nMOS) são chamados circuitos CMOS (*Complementary Metal Oxide Semiconductor*). Esses circuitos são mais sensíveis a colisões de partículas de alta energia, se comparados aos circuitos formados por transistores BJT.

A Figura 2 mostra de forma esquemática a vista seccionada de um transistor MOSFET. O transistor representado possui substrato formado por material semicondutor dopado do tipo p (nMOS), a título de exemplificação. A explicação para transistores com substrato do tipo n (pMOS) é análoga.

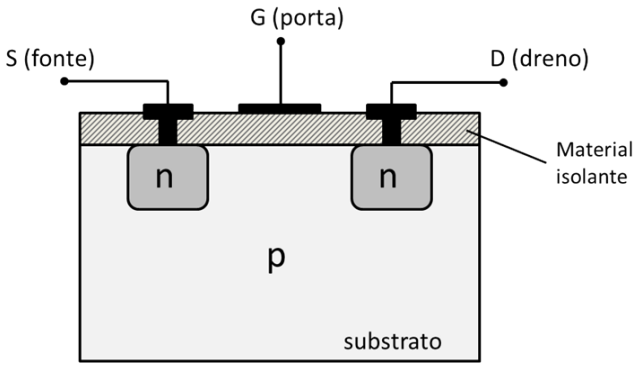


Figura 2 – Representação de um transistor MOSFET.

Na operação do transistor representado, a tensão aplicada à porta (V_{GS}) controla a corrente de elétrons que vai da fonte para o dreno ou vice-versa.

Quando $V_{GS} = 0$, o substrato do tipo p não permite a passagem de corrente entre os terminais de fonte e dreno conectados ao material n-dopado, devido à camada de depleção entre os materiais n e p.

A Figura 3 representa o que ocorre quando uma tensão positiva é aplicada à porta.

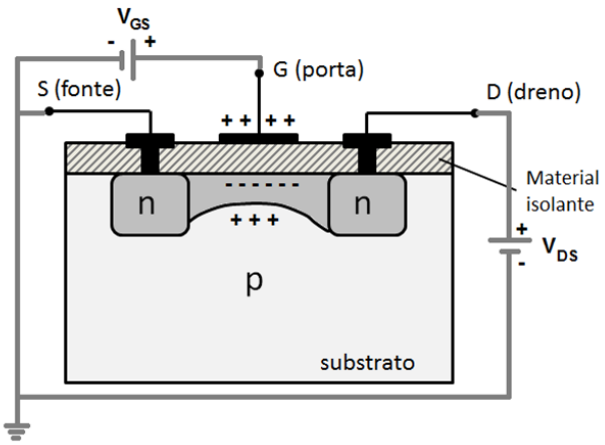


Figura 3 – Representação de um transistor MOSFET com $V_{GS} > V_{DS}$.

Nesse caso, a carga positiva no terminal da porta repele as lacunas (portadoras de carga positiva) criando um caminho para que a corrente de elétrons flua da fonte para o dreno.

Quando partículas ionizadas de alta energia atingem a região próxima à junção dos transistores em um circuito CMOS, portadores minoritários são criados, tal como representado na Figura 4.

Se o choque das partículas for dotado de energia suficiente[1] para que esses portadores sejam coletados pelas regiões de difusão dos terminais de fonte e dreno, haverá uma alteração na carga desses terminais. Havendo um acúmulo de carga além de um determinado limite, denominado *Carga Crítica*, haverá indução de corrente na junção p-n, registrando-se sinais incorretos nesse transistor (WANG; AGRAWAL, 2008). A esse fenômeno dá-se o nome de *Single Event Effect* (SEE).

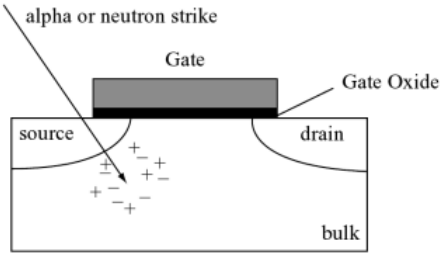


Figura 4 – Interação de uma partícula de alta energia com o substrato em um circuito CMOS.

Fonte: (MUKHERJEE, 2011, p. 26)

SEEs podem ocorrer em diversas regiões de um circuito. Se um SEE ocorre em um circuito combinacional, por exemplo, seu efeito é transitório. Nesse caso denomina-se *Single Event Transient* (SET) (ANDJELKOVIC et al., 2015). Os SEEs podem ocorrer, no entanto, em células de memória. Nesse caso tais eventos são denominados *Single Event Upset* (SEU) (DODD; MASSENGILL, 2003).

Falhas do tipo SEU podem ocorrer em regiões de memória de interesse a uma dada aplicação, causando mal funcionamento. Os SEU são, portanto, o tipo de falha de interesse ao longo desse trabalho.

2.2 Histórico

O primeiro trabalho a mencionar problemas relacionados a SEE em circuitos integrados não foi um trabalho a respeito de eletrônica em aplicações espaciais, mas um trabalho de 1962 sobre tendências de escala na microeletrônica em aplicações terrestres (WALLMARK; MARCUS, 1962). Nesse trabalho os autores alertam para uma limitação à densidade de integração de semicondutores, alegando que a tecnologia da época já estava próxima ao limite imposto pelo bombardeamento de raios cósmicos na superfície da Terra.

[1] Essa energia é medida por uma grandeza denominada LET (*Linear Energy Transfer*), que é a medida da energia transferida pela partícula ionizadora ao material do semicondutor em função do comprimento de sua trajetória.

Em 1975, Binder publicou um artigo reportando upsets registrados em um satélite de comunicação (BINDER; SMITH; HOLMAN, 1975). Ao longo de 17 anos de operação foram registrados 4 eventos em flip-flops do tipo J-K formados por transistores de junção bipolar.

Em 1978, May e Woods, da *Intel Corporation*, observaram efeitos aleatórios de inversão de bits em chips de memória fabricados pela empresa. Eles publicaram uma análise em 1979 (MAY; WOODS, 1979), onde diagnosticaram o problema como resultado da ação de partículas alfa sobre os circuitos. Foi identificada uma contaminação por urânio durante o processo de encapsulamento desses circuitos, proveniente de uma região próxima ao local onde o encapsulamento era feito.

Ao final da década de 70 um maior número de erros devido a raios cósmicos foi identificado em circuitos CMOS, e foram feitas as primeiras tentativas de previsão da taxa de falhas (PICKEL; JR, 1978). O termo *Single Event Upset* foi utilizado por Guenzer pela primeira vez em 1979 (GUENZER; WOLICKI; ALLAS, 1979). Nesse ano também foi reportado o primeiro caso de *Single Event Latchup* (KOLASINSKI et al., 1979), fenômeno no qual efeitos permanentes geram mal funcionamento do circuito podendo causar o seu colapso devido à sobrecorrente gerada. Embora esse efeito esteja fora do escopo desse trabalho, essa descoberta indicava o efeito crescente da ação dos raios cósmicos sobre os circuitos integrados.

Na década de 80 os estudos relacionados a SEE tiveram continuidade à medida em que casos de erros observados eram reportados. Entre 1986 e 1987 a IBM identificou um problema semelhante ao divulgado por May e Woods, da Intel. A empresa teve seus *chips* contaminados por radiação, e atribuiu o ocorrido a uma indústria química que utilizava material radioativo no processo de limpeza de recipientes. Esses recipientes eram utilizados para armazenar um ácido usado no processo de fabricação dos circuitos (MUKHERJEE, 2011).

Em 1995, Baumann (BAUMANN et al., 1995) identificou isótopos de Boro, que anteriormente era utilizado na dopagem de semicondutores do tipo n, como uma fonte causadora de SEE devido a sua instabilidade quando expostos a nêutrons. Posteriormente, em 1996, Normand reportou observações de efeitos dos raios cósmicos ao estudar logs de erros em sistemas computacionais (NORMAND, 1996)

Na década seguinte, apesar de haver pouca divulgação, foram conhecidos alguns casos de SEE em sistemas comerciais (MUKHERJEE, 2011). Em 2000, a *Sun Microsystems* relatou problemas em servidores baseados no *UltraSPARC-II*. Alguns incidentes também foram relatados pela *Cypress semiconductor* em 2004. Em 2005 a *Hewlett-Packard* relatou problemas frequentes com efeitos de raios cósmicos em servidores do Laboratório Nacional de Los Alamos, situado a mais de 2000 m de altitude em relação ao nível do mar.

Atualmente os efeitos de partículas de alta energia em circuitos integrados constituem um assunto de interesse crescente na comunidade científica (VELAZCO; PERONNARD;

HUBERT, 2009; CHEMINET et al., 2013; VARGAS et al., 2015). A tendência é que, com o aumento de aplicações espaciais e da densidade de material semicondutor nos circuitos utilizados nessas aplicações, a busca por formas de mitigação para esses problemas continue em direção a novas soluções.

2.3 Principais fontes de Single Event Effects

Os SEEs provenientes da radiação no espaço são causados, basicamente, por partículas alfa ou nêutrons (MUKHERJEE, 2011).

As partículas alfa são formadas por um conjunto de dois prótons e dois nêutrons, unidos em uma estrutura equivalente ao núcleo de um átomo de hélio. Já os nêutrons provém da quebra de átomos.

Partículas alfa e nêutrons interagem de maneiras diferentes com os circuitos CMOS. No caso das partículas alfa, a criação de pares elétron-lacuna ocorre devido ao depósito de cargas ao longo da trilha de ionização criada por essa partícula à medida em que vai adentrando o substrato semicondutor (PETERSEN, 1983). Já os nêutrons atuam nos circuitos CMOS por meio de colisões, elásticas e inelásticas. No primeiro caso, as partículas mantêm sua identidade, ao passo que no segundo caso cria-se partículas secundárias na colisão (MUKHERJEE, 2011).

Tanto as partículas alfa como os nêutrons proveem de fontes de radiação. Na baixa órbita terrestre, os circuitos CMOS sofrem o efeito primário dessas fontes, os chamados *Raios Cósmicos Primários*. À medida em que esses raios atingem a atmosfera terrestre, é disparada uma reação em cascata, como mostrado na Figura 5, dando origem a outras partículas, tais como píons, múons e nêutrons (NICOLAIDIS, 2010; MUKHERJEE, 2011). Esse novo conjunto de partículas constitui os *Raios Cósmicos Secundários*. Parte das partículas geradas nos raios cósmicos secundários possui decaimento muito rápido. As partículas que conseguem chegar à superfície da Terra constituem os *Raios Cósmicos Terrestres*.

De acordo com Ziegler e Lanford (ZIEGLER; LANFORD, 1979), os efeitos dos raios cósmicos na superfície terrestre são pouco frequentes se comparados aos efeitos dos raios primários do espaço. Estima-se que os raios cósmicos terrestres são formados por menos de 1 % das partículas primárias (MUKHERJEE, 2011, p. 24).

Os raios cósmicos primários, assunto de interesse nesse trabalho, basicamente se dividem em dois tipos - *Partículas Galácticas* e *Partículas Solares* (MUKHERJEE, 2011). As partículas galácticas são emitidas por explosões de supernovas, rajadas provenientes de outras estrelas e reações que ocorrem nas regiões entre galáxias. Tais partículas geralmente possuem energias superiores a $1GeV$ [2], e viajam no espaço em alta velocidade. Um

[2] A unidade de medida elétron-Volt, utilizada para expressar energia de partículas, equivale ao trabalho realizado por um elétron acelerado por um potencial de $1V$

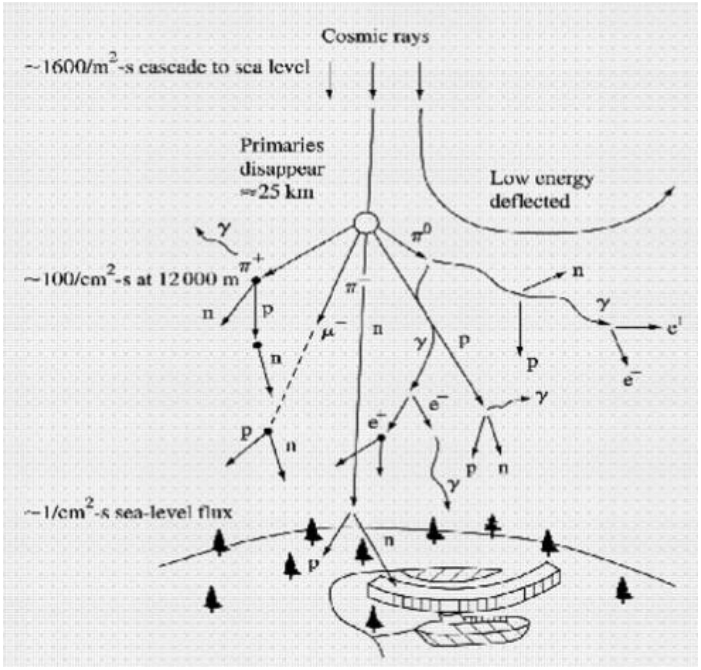


Figura 5 – Colisões de raios cósmicos na atmosfera terrestre.

Fonte: (ZIEGLER et al., 1996)

exemplo de um fenômeno observado no universo que envole grande quantidade de energia consiste no colapso de estrelas massivas. Esse fenômeno, denominado *Gamma Ray Burst*, é uma fonte intensa de partículas galácticas provenientes de radiação e vem sendo bastante estudado pela comunidade científica (COOK et al., 1993; PARSONS, 2001; SCHANNE et al., 2005).

As partículas solares, por sua vez, têm sua origem nas atividades e erupções do Sol. Essas partículas possuem energia inferior se comparadas às partículas galácticas mas, devido à proximidade do Sol em relação à Terra, são constantemente estudadas, como mostrado na Figura 6, que mostra os satélites que compõem o *Heliophysics System Observatory* (HSO) da NASA (*National Aeronautics and Space Administration*) para estudo das interações das erupções solares com a Terra.

A energia das partículas provenientes de erupções solares depende do ciclo solar. A atividade do sol é caracterizada por certa periodicidade, de forma que observa-se picos de erupções solares a cada 11 anos (BOUDENOT, 2007).

Pode-se analisar a atividade solar observando-se as manchas formadas em sua superfície. Essas observações começaram a ser feitas por volta de 1610 por Galileu Galilei. A Figura 7[3] mostra a média mensal de manchas solares observadas desde 1750 até os dias de hoje.

[3] Disponível em: http://solarscience.msfc.nasa.gov/SunspotCycle.shtml
Data de acesso: 21 de Dezembro de 2015.

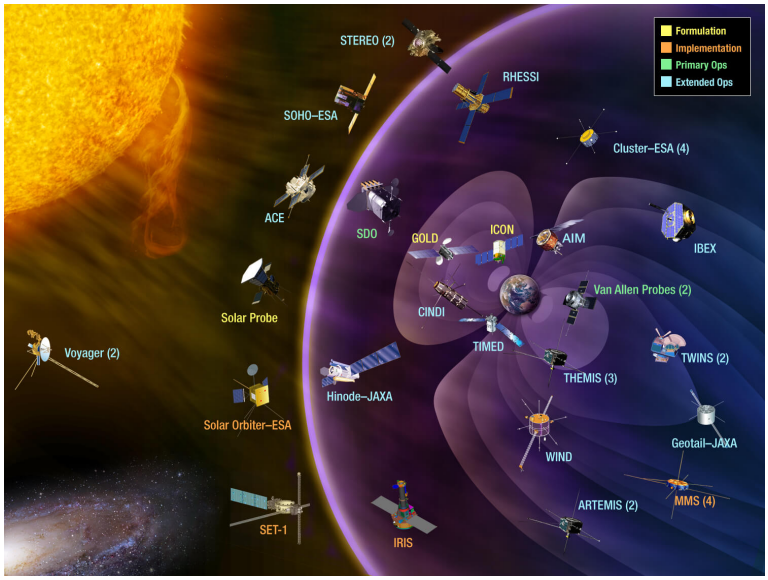


Figura 6 – Concepção artística dos satélites envolvidos no estudo de erupções solares.

Fonte: Heliophysics System Observatory (HSO) - NASA

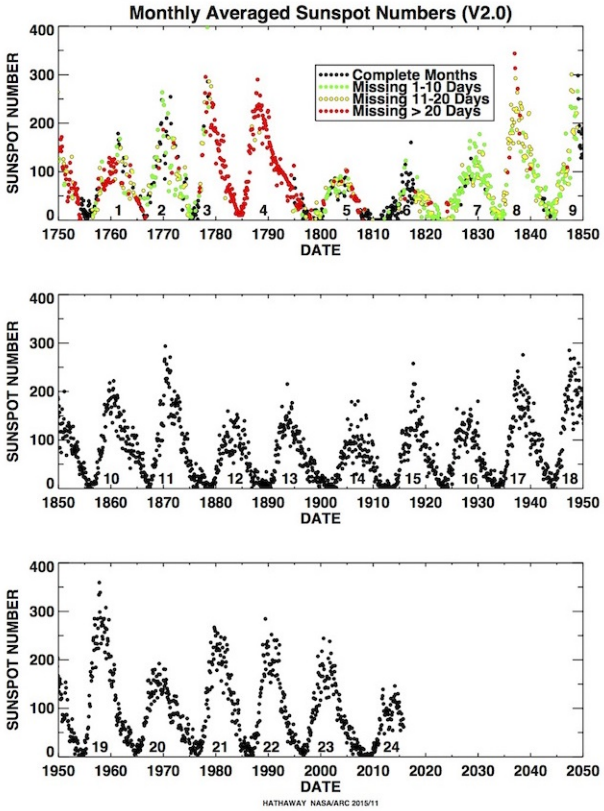


Figura 7 – Observações de manchas solares ao longo dos anos.

Fonte: Marshall Space Flight Center - NASA

Os efeitos da atividade solar podem impactar o funcionamento de circuitos integrados tanto em órbita terrestre quanto em solo. Em um episódio recente, as atividades de tráfego aéreo foram interrompidas por cerca de uma hora na Suécia, no dia 4 de novembro de 2015, devido a tempestades solares ocorridas no dia (ver Anexo A).

2.4 Os Cinturões de Van Allen e a Anomalia Magnética do Atlântico Sul

É um fato conhecido que o campo magnético da Terra, denominado *Campo Geomagnético*, interage com as partículas de radiação provenientes do espaço, defletindo parte dessas partículas (STASSINOPOULOS; RAYMOND, 1988).

As linhas de campo relativas ao campo geomagnético assumiriam um formato de dipolo caso não houvessem os ventos solares. No entanto, na presença desses ventos ocorre uma distorção das linhas de campo, que assumem um formato de parábolas, tal como mostrado na Figura 8.

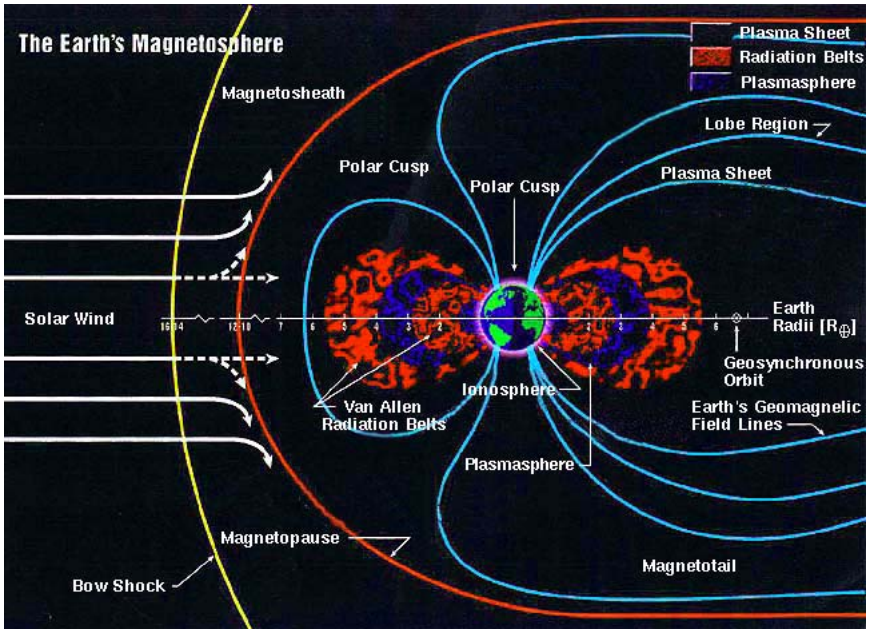


Figura 8 – Efeito dos ventos solares sobre o campo geomagnético.

Fonte: Arizona State University - Dr. Holbert's EEE 598 Course

Na interação das partículas carregadas provenientes de ventos solares com o campo magnético da Terra, uma parte dessas partículas é repelida, outra parte consegue atravessar o campo chegando à atmosfera terrestre. No entanto, algumas dessas partículas podem ser "capturadas"por esse campo, passando a movimentar-se ao seu redor (LUI,

2000). Esse movimento pode ser explicado pelas equações que descrevem uma carga elétrica em movimento.

Seja uma partícula de massa m e carga elétrica q se movendo com velocidade $\mathbf{v}$ na presença de um campo magnético $\mathbf{B}$. A equação que descreve o movimento dessa partícula no tempo é dada por

$$m\frac{d\mathbf{v}}{dt} = q\mathbf{v} \times \mathbf{B}. \tag{1}$$

Decompondo-se a velocidade $\mathbf{v}$ em duas componentes, uma delas perpendicular à linha do campo magnético $\mathbf{B}$, dada por $\mathbf{v}_\perp$, e outra paralela à linha de campo, definida por $\mathbf{v}_{||}$, temos

$$\frac{d\mathbf{v}_{||}}{dt} = 0; \qquad \frac{d\mathbf{v}_\perp}{dt} = \frac{q}{m}\mathbf{v}_\perp \times \mathbf{B}. \tag{2}$$

A Equação 2 indica que a velocidade da partícula na direção paralela ao campo magnético é constante, ao passo que a velocidade da componente perpendicular é circular, com frequência angular dada por $\frac{q}{m}B$. Em outras palavras, a partícula tende a acompanhar as linhas de campo, movendo-se de forma circular em torno dessas linhas, tal como mostrado na Figura 9.

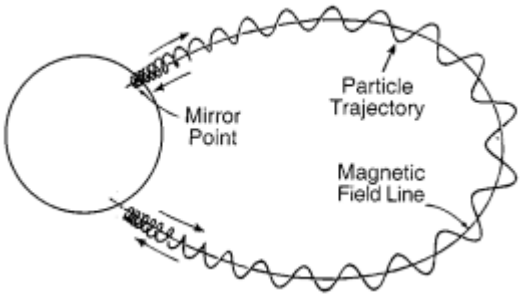


Figura 9 – Movimento de uma partícula carregada em torno das linhas do campo geomagnético.

Fonte: (LUI, 2000)

A captura das partículas carregadas em torno das linhas do campo geomagnético formam cinturões de partículas radioativas, denominados *Cinturões de Van Allen* (Figura 10). Esses cinturões são formados por dois anéis, sendo que os *cubesats* geralmente orbitam regiões próximas aos anéis internos.

Outro fator que influencia a distribuição de partículas de alta energia é a inclinação da Terra em relação ao eixo do campo geomagnético. Essa inclinação gera uma perturbação nos cinturões devido ao enfraquecimento do campo magnético em uma dada região, resultando em um acúmulo maior de partículas radioativas (CILLIERS; OPPERMAN; MEYER, 2009).

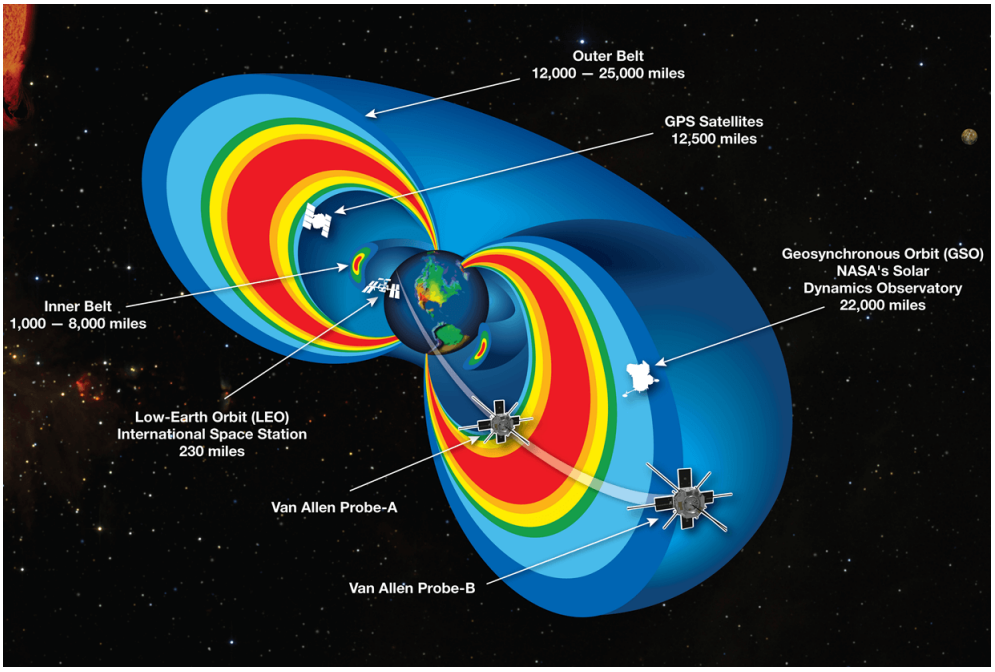


Figura 10 – Cinturões de Van Allen. Nessa figura pode-se observar a posição da órbita de alguns artefatos de tecnologia espacial em relação aos cinturões. A Estação Espacial Internacional (ISS), por exemplo, orbita em proximidade ao cinturão interno.

Fonte: "Space Radiation Effects on Microelectronics", NASA Jet Propulsion Laboratory

A região em questão cobre o sul do Brasil e parte do Oceano Atlântico, sendo denominada *Anomalia Magnética do Atlântico Sul*, ou simplesmente SAA (do inglês, *South Atlantic Anomaly*). A região do SAA está destacada na Figura 11.

Em 30 de dezembro de 1995 a NASA lançou o RXTE (*Rossi X-Ray Timing Explorer*), um satélite com instrumentação adequada ao estudo de partículas de alta energia ao redor da Terra (SMITH et al., 1996). Um dos instrumentos presentes do RXTE foi o HEXTE (*High Energy X-Ray Timing Experiment*), que continha um monitor de partículas voltado para detectar o aumento no fluxo de partículas de alta energia à medida em que o satélite sobrevoasse a região da SAA.

A Figura 12 mostra um levantamento da escala de fluxo nas bordas da região da SAA. Pode-se observar que o fluxo de partículas de alta energia na região central da anomalia é significativamente superior às regiões da borda - trata-se de um fluxo cerca de 1000 vezes mais intenso no centro da anomalia em relação às extremidades.

Conforme mencionado na Seção 1.3, a localização do Brasil em uma região de maior densidade de partículas de alta energia reforça a importância de se desenvolver sistemas que possam operar com maior robustez ao efeitos dessas partículas.

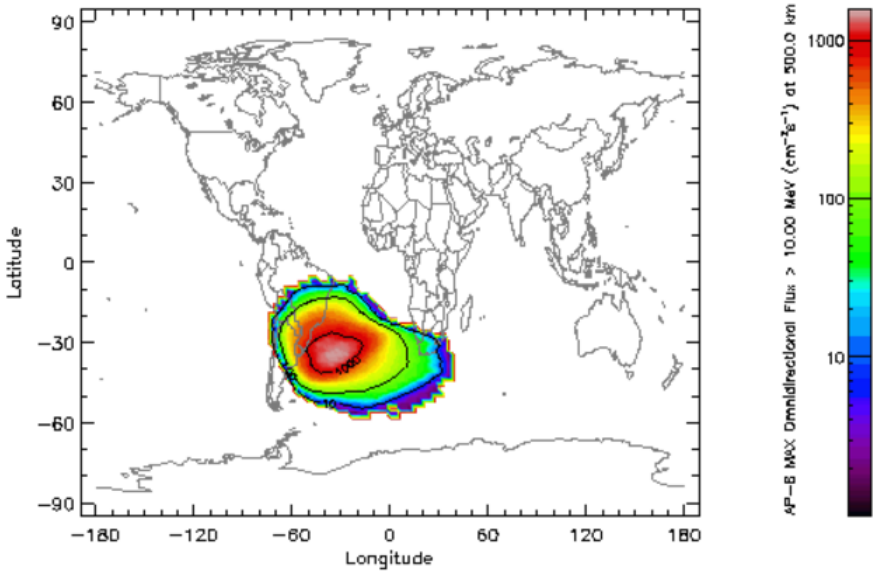


Figura 11 – Região da Anomalia Magnética do Atlântico Sul. A escala de cores indica o fluxo de nêutrons na região.

Fonte: (RUANO; MAESTRO; REYES, 2007)

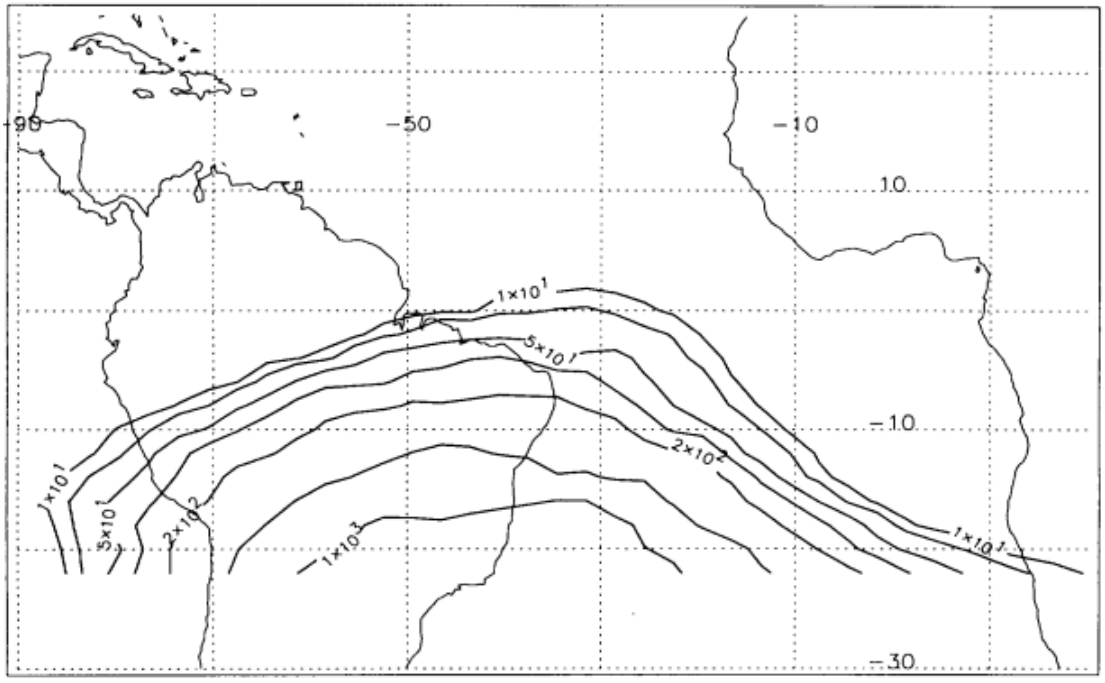


Figura 12 – Contornos da região da Anomalia Magnética do Atlântico Sul levantados pelo HEXTE.

Fonte: (SMITH et al., 1996)

2.5 Formas de Mitigação

Existem diferentes formas de mitigação de falhas do tipo SEU em sistemas embarcados. Uma abordagem se resume à blindagem (*Shielding*) do encapsulamento do circuito. Para essa solução o maior desafio consiste na definição de um material que ofereça uma blindagem apropriada sem tornar o peso do circuito uma limitação à aplicação. Recentemente a blindagem eletrostática tem sido pesquisada na solução desse problema (TRIPATHI; WILSON; YOUNGQUIST, 2006).

Alternativamente, outra forma de mitigação consiste no monitoramento dos efeitos de radiação nos circuitos, permitindo a identificação de alguma alteração indevida de tensão ou corrente (KOZAK; MAKOWSKI; NAPIERALSKI, 2012). Apesar de permitir um controle preciso desses efeitos, essa solução aumenta a complexidade e o custo da aplicação.

As formas de mitigação mais presentes na literatura se baseiam em redundância nos diferentes níveis:

- *Hardware* - consiste na criação de réplicas de componentes cuja função é a mesma no circuito. Algumas aplicações possuem elementos reconfiguráveis que contribuem para prover maior disponibilidade do sistema (ZHOU; YANG; WANG, 2008);
- *Software* - a redundância de software prevê desenvolvimento dissimilar para uma mesma função. A ideia é ter a mesma funcionalidade em regiões distintas da memória, diminuindo a probabilidade de erros devido ao efeito das partículas de alta energia (HOLT, 1997);
- *Informação* - semelhante à redundância de software, porém mais focada na informação salva. Nesse caso a mesma informação é armazenada em diversas regiões da memória (GRECKI; JABLONSKI; MAKOWSKI, 2009);
- *Tempo* - a redundância temporal consiste no processamento de uma função mais de uma vez, realizando votações e consolidando o resultado final. Esse método exige o escalonamento apropriado das tarefas (JACOBS; WULF; GEORGE, 2013).

Raramente uma solução irá se basear em apenas uma forma de redundância, visto que a aplicação integra características afetadas por todos os métodos citados. Dessa forma é comum encontrar soluções com tolerância a falhas baseadas em uma mescla dos tipos de redundância citados. O SDATF, a ser descrito no próximo capítulo, é um exemplo de solução híbrida.

Capítulo 3

Sistema de Determinação de Atitude com Tolerante a Falhas

Nos capítulos anteriores foi introduzida a proposta do trabalho que aqui se apresenta, bem como os conceitos teóricos relacionados ao tipo de falhas que são estudadas nesse trabalho. Nesse capítulo será descrito o Sistema de Determinação de Atitude com Tolerância a Falhas (SDATF), objeto de estudo na modelagem e validação, sua funcionalidade e o contexto em que se encontra, como carga útil do satélite NanosatC-Br2.

3.1 NanosatC-Br2

O lançamento do NanosatC-Br ocorreu no dia 19 de junho de 2014, na base aérea de Dombarovsky, unidade federativa Oblast de Oremburgo na Rússia. Três cargas úteis foram embarcadas nesse cubsat - um magnetômetro de três eixos, cujo objetivo era a coleta de dados referentes ao campo geomagnético, especialmente os distúrbios da Anomalia Magnética do Atlântico Sul; um circuito integrado projetado pela *Santa Maria Design House* da UFSM; e o hardware FPGA, projetado para suportar efeitos da radiação no espaço.

O NanosatC-Br2 deve dar continuidade ao programa, levando cargas úteis de interesse da comunidade científica e ao desenvolvimento tecnológico. O primeiro satélite do programa NanosatC-Br foi do tipo 1U, ou seja, formado por uma estrutura modular cúbica, ao passo que o NanosatC-Br2 é um satélite 2U, constituído por dois módulos cúbicos acoplados. A Figura Figura 13 mostra o modelo de engenharia do NanosatC-Br2 em testes no Laboratório de Simulação (LABSIM), nas dependências do INPE em São José dos Campos (SP).

A composição 2U do NanosatC-Br2 prevê um dos módulos dedicados às cargas úteis (além de uma bobina de torque), o que aumenta as possibilidades em relação ao primeiro cubesat. Entre as cargas úteis previstas estão dois magnetômetros para estudo do campo

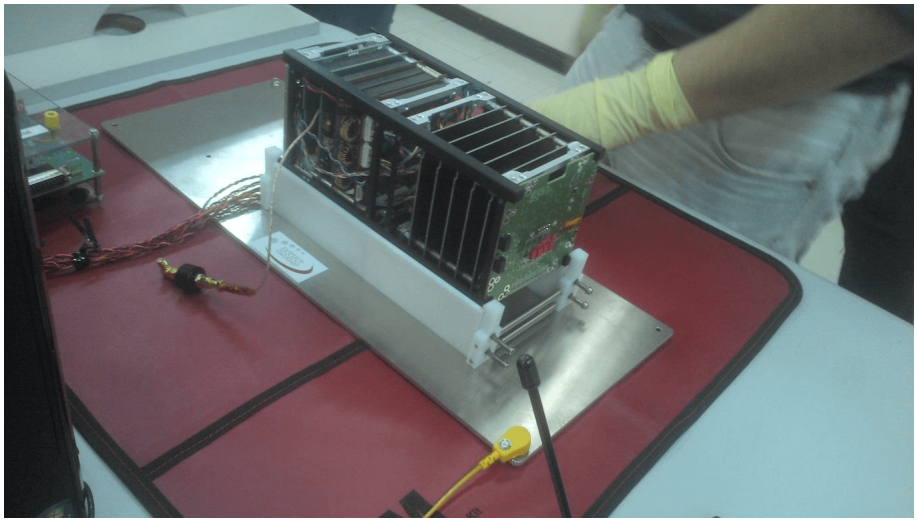


Figura 13 – Modelo de engenharia do NanosatC-Br2.

Fonte: http://brazilianspace.blogspot.com.br/ - Acessado em 21 de Dezembro de 2015.

geomagnético, uma *Sonda de Langmuir*[1] desenvolvida pelo INPE, um FPGA tolerante a falhas por radiação, desenvolvido pela Universidade Federal do Rio Grande do Sul (UFRGS), um circuito integrado com proteção contra radiação, desenvolvido pela Santa Maria Design House, e o SDATF, objeto de estudo nesse trabalho.

3.2 Determinação de Atitude

A atitude de um dado corpo se refere a sua orientação no espaço. A atitude de um satélite, portanto, representa a sua rotação em torno do seu centro de massa.

Existem diferentes formas de se representar a atitude de um dado corpo (DIEBEL, 2006). Uma forma comum é a utilização dos *Ângulos de Euler*, cuja representação consiste em uma matriz definida em função de três rotações consecutivas, pelos ângulos ψ, θ e ϕ, em torno dos eixos cartesianos do espaço tridimensional z, x e y, respectivamente. Trata-se de uma representação aparentemente simples, no entanto a descrição da dinâmica de sistemas por meio dos ângulos de Euler não é trivial.

Uma representação alternativa aos ângulos de Euler para a atitude de um corpo são os chamados quatérnios (BIASI; GATTASS, 2007). Um quatérnio consiste na generalização de um número complexo para o espaço tridimensional, uma vez que o número complexo é definido no plano bidimensional.

Um quatérnio q pode ser representado por

[1] A Sonda de Langmuir serve para fazer medições da densidade do plasma ionosférico ao longo do trajeto do cubesat. O plasma ionosférico é gerado a partir de raios ultra-violeta, raios X e íons provenientes do Sol.

$$q = a + b\hat{\mathbf{i}} + c\hat{\mathbf{j}} + d\hat{\mathbf{k}} \tag{3}$$

onde a, b, c e d são números reais, parâmetros do quatérnio, e $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$ e $\hat{\mathbf{k}}$ são versores unitários definidos tais que

$$i^2 = j^2 = k^2 = -1. \tag{4}$$

Uma descrição detalhada de como os quatérnios podem ser utilizados para representar a atitude de um satélite foge ao escopo desse trabalho. Para os fins dessa dissertação é suficiente asseverar que os quatérnios possuem propriedades convenientes aos cálculos envolvidos na determinação da atitude (BIASI; GATTASS, 2007).

A atitude de um satélite não é diretamente medida por algum tipo de sensor. Em geral, a determinação de atitude em três eixos é feita de forma indireta com base na medição de dois sensores, por meio de algum algoritmo de fusão sensorial (HALL, 2003, ch. 4, p. 2).

Usualmente sensores solares e magnetômetros são empregados em aplicações de determinação de atitude em satélites. Os sensores solares fornecem um vetor, com origem no centro de referência do corpo, que aponta em direção ao sol. O magnetômetro fornece um vetor que representa a intensidade do campo magnético da Terra na direção dos três eixos.

Existem diversos algoritmos que utilizam as medições desses sensores e, utilizando modelos de referência da posição do Sol e do campo magnético da Terra, estimam a atitude do satélite. A determinação de atitude no SDATF usa o método QUEST (*Quaternion Estimator*).

O QUEST (SHUSTER; OH, 1981) se enquadra numa categoria de algoritmos que buscam determinar o quatérnio de atitude por meio de otimização. O QUEST é uma derivação computacionalmente simplificada de outro método de determinação de atitude, denominado *Método q* (FERREIRA et al., 2008).

O Método q é baseado em dois ou mais vetores de observação w_i, com $i = 1, 2, ..., n$. Para cada vetor de observação deve-se ter um vetor de referência v_i. O método consiste na busca pela matriz de rotação A tal que

$$w_i = Av_i. \tag{5}$$

Para o caso da determinação de atitude em um satélite que utiliza sensor solar e magnetômetro, temos w_1 e w_2 obtidos por meio das medições desses sensores. Os vetores de referência v_1 e v_2 são obtidos de modelos de órbita e do campo magnético da Terra.

Os algoritmos baseados em otimização como o Método q e o QUEST buscam, de forma iterativa, uma matriz A que minimiza o erro para os vetores de observação.

Detalhes da implementação do método QUEST fogem a definição dos requisitos essenciais para definição da solução com tolerância a falhas do SDATF. Mais detalhes podem ser encontrados em (SHUSTER; OH, 1981) e (SHUSTER, 2004).

3.3 SDATF: Descrição da Solução Tolerante a Falhas

As seções 3.1 e 3.2 serviram apenas como contextualização do SDATF, sistema que será estudado em detalhes de agora em diante. Nessa seção seu comportamento será descrito, e nos próximos capítulos o processo de modelagem e validação será apresentado.

A Figura 14 mostra a arquitetura do Sistema de Determinação de Atitude com Tolerância a Falhas.

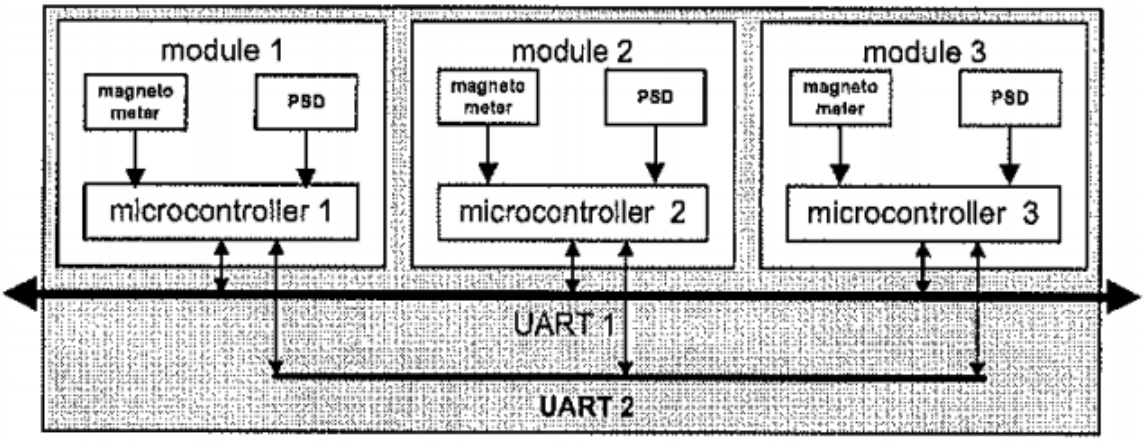


Figura 14 – Arquitetura do SDATF.

Fonte: (DUARTE R. O.; TORRES; KUGA, 2011)

Essa arquitetura é representada por três módulos em redundância. Cada módulo possui um microcontrolador com comunicação com dois sensores - um PSD (*Position Sensitive Detector*) usado como sensor solar e um magnetômetro de três eixos.

A comunicação entre os microcontroladores se dá por meio de um canal UART (*Universal Asynchronous Receiver/Transmitter*), e o outro canal é utilizado para a comunicação com o computador de bordo do satélite.

Nessa aplicação são utilizados três microcontroladores de mesma arquitetura, programados com o mesmo software. Cada microcontrolador pode assumir um dos seguintes papéis - MASTER, SAMPLER ou SLEEPER.

No modo MASTER, o microcontrolador atua como interface entre o sistema de determinação de atitude e o computador de bordo do satélite. O computador de bordo deve ser visto como um elemento externo ao SDATF - ele inicializa o sistema e deve receber,

periodicamente, a informação de atitude calculada pelo SDATF. Ao longo desse trabalho o computador de bordo será referenciado como OBC, da sigla *OnBoard Computer*.

No modo MASTER, portanto, o microcontrolador coordena a operação dos demais, valida os dados medidos pelo SAMPLER e fornece periodicamente a informação consolidada por meio de uma interface ao OBC.

No modo SAMPLER, o microcontrolador deve amostrar os sensores sob demanda do MASTER, calcular a atitude, e fornecer essa informação dentro de um período estabelecido. Ele também monitora o funcionamento do MASTER, devendo assumir essa posição caso não receba dentro de um limite de tempo um sinal de monitoramento de funcionamento do MASTER (*hearbeat*).

No modo SLEEPER o microcontrolador se encontra em um estado de baixo consumo de energia, devendo assumir o papel de MASTER quando inicializado pelo computador de bordo, ou de SAMPLER, quando inicializado por outro microcontrolador.

Em um momento inicial, os três microcontroladores assumem o papel de SLEEPER, até que o computador de bordo inicialize um deles, por meio de uma interrupção, para assumir o papel de MASTER.

Ao ser inicializado, o MASTER deve "acordar" o microcontrolador vizinho para assumir o papel de SAMPLER, por meio de uma interrupção, e enviar um sinal de *acknowledgement* para o computador de bordo quando houver sucesso em sua inicialização e do microcontrolador vizinho. Caso contrário, o computador de bordo deve inicializar outro microcontrolador. Antes de emitir o sinal de interrupção ao outro microcontrolador, o MASTER amostra os sensores, calcula a atitude e salva essa informação como última atitude válida.

Uma vez estabelecida a comunicação entre MASTER e SAMPLER, ocorre o envio escalonado de informações entre eles, sincronizados dentro de um *frame* com duração de 1 segundo, tempo esperado para que a atitude seja enviada ao computador de bordo do satélite. O escalonamento ocorre conforme a seguinte lógica:

- O MASTER envia periodicamente um sinal reconhecido pelo SAMPLER, um inteiro incremental denominado *heartbeat*. Caso o SAMPLER não receba esse sinal a tempo ou receba um valor incorreto, ele envia um sinal de reset ao MASTER e assume o seu papel, inicializando o outro microcontrolador para assumir o papel de SAMPLER;

- Concomitantemente, enquanto não recebe esse sinal, o SAMPLER deve amostrar os sensores, calcular a atitude e enviar ao MASTER dentro de um limite de tempo estabelecido. Caso esse limite de tempo não seja respeitado, o MASTER deve enviar um sinal de reset ao SAMPLER e inicializar outro microcontrolador para assumir o papel;

- Uma vez que o MASTER recebe a atitude calculada pelo SAMPLER, ele deve validar esse sinal, comparando-o com a atitude previamente calculada. Se os valores

forem compatíveis, a atitude consolidada é armazenada como última atitude válida e referência para a próxima comparação. Caso os valores não sejam compatíveis, o MASTER incrementa um contador que representa o número de vezes em que o SAMPLER enviou atitude inválida. Após receber valor inválido para atitude pela segunda vez, o MASTER envia um sinal de reset ao SAMPLER e inicializa o outro microcontrolador;

- ❏ Após enviar a atitude ao MASTER pela terceira vez, o SAMPLER aguarda o início de um novo *frame*. Após o envio do sinal de *heartbeat* pela quarta vez, o MASTER deve enviar a última atitude válida ao computador de bordo do satélite e aguardar pelo início de um novo *frame*.

- ❏ Ao término de 1 segundo, os contadores e temporizadores são inicializados e o processo se repete.

Esse processo se repete enquanto o sistema estiver em operação. O chaveamento de papéis entre os três microcontroladores deve garantir o envio de atitude válida ao computador de bordo periodicamente.

A solução tolerante a falhas proposta para o SDATF possui características de técnicas conhecidas na literatura, como Redundância Modular Tripla (LYONS; VANDERKULK, 1962), definida nos três módulos idênticos, *Cold stand-by spare* (ROBINSON; NEUTS, 1989), representado pelo microcontrolador aguardando no estado SLEEPER, e controle de vivência mútua (CHUNBO; YANLIN, 2010), característica na qual MASTER e SAMPLER monitoram um ao outro por meio do envio da atitude e do *heartbeat*.

A operação do SDATF é baseada em um escalonamento de tarefas entre MASTER e SAMPLER dentro de um *frame* de 1 segundo. Esse escalonamento é implementado por meio de *timers* programados em cada agente. Para o escalonamento, os microcontroladores possuem três *timers* programados, sendo que os valores e o significado desses *timers* varia com o papel exercido pelo microcontrolador em um dado momento.

A Figura 15 mostra o escalonamento entre MASTER e SAMPLER baseado nos temporizadores.

Para o MASTER, o estouro do *timer* 1, de 220 ms, indica que o sinal de *heartbeat* deve ser enviado ao SAMPLER, e o *timer* 2, de 280 ms é o tempo máximo de espera pelo envio da atitude por parte do SAMPLER. Quando o *heartbeat* é enviado ou a atitude é recebida os *timers* 1 e 2, respectivamente são inicializados e disparados novamente.

Para o SAMPLER, o *timer* 1, de 250 ms, é o tempo máximo aguardado pelo envio do *heartbeat* por parte do MASTER, e o estouro *timer* 2, de 240 ms, indica que o SAMPLER deve enviar a atitude calculada ao MASTER. Assim como no caso do MASTER, os *timers* são inicializados e disparados novamente, para o *timer* 1 quando o *heartbeat* é recebido, e para o *timer* 2 quando a atitude é enviada.

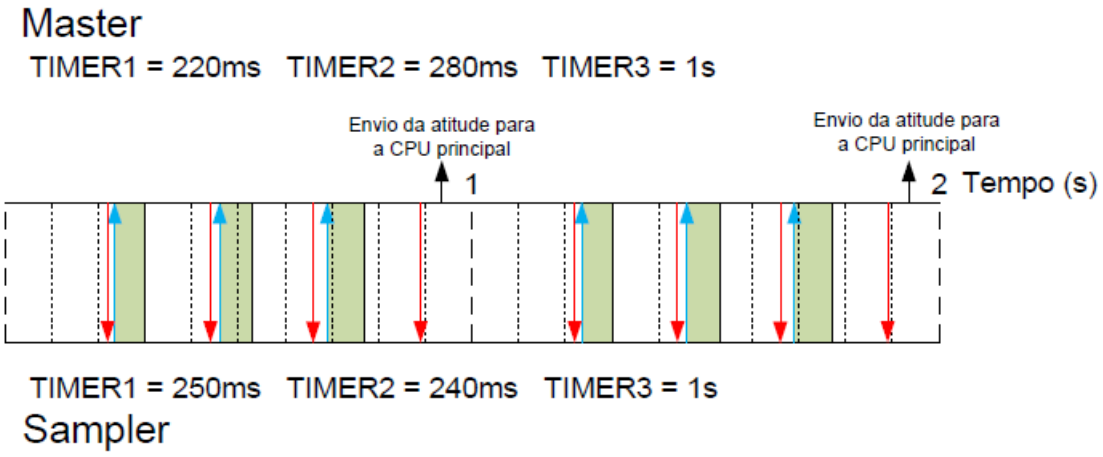


Figura 15 – Escalonamento entre MASTER e SAMPLER.

Fonte: (TORRES, 2013)

Tanto no caso do MASTER quanto no caso do SAMPLER, o *timer* 3 sinaliza o término do *frame* de sincronização. Na Figura 15 as setas em vermelho (de cima para baixo) indicam o envio do *heartbeat* do MASTER para o SAMPLER, e as setas em azul (de baixo para cima) indicam o envio da atitude do SAMPLER para o MASTER. As linhas pontilhadas a cada 100 ms estabelecem a escala temporal do diagrama, e a área hachurada (em verde) indica o período no qual o MASTER valida a atitude enviada pelo SAMPLER. Essa temporização tem por objetivo coordenar a operação em monitoramento mútuo evitando a colisão de dados entre MASTER e SAMPLER.

A descrição do comportamento do SDATF foi feita nessa seção em linguagem natural. Embora uma descrição dessa forma seja de fácil compreensão, a tarefa de se criar um modelo de simulação que represente esse comportamento de forma fiel não é uma tarefa simples, visto que há um grande número de possibilidades de tarefas sendo executadas por cada microcontrolador em um dado momento. Com esse intuito, uma abordagem adequada consiste na criação de uma descrição intermediária, baseada em um formalismo que permita descrever esse comportamento em uma linguagem controlada. Esse formalismo será apresentado no próximo capítulo.

Capítulo 4

Validação do Sistema Tolerante a Falhas baseada em Modelagem

Nos capítulos anteriores contextualizou-se a ocorrência de falhas do tipo *Single Event Upset* durante a operação de circuitos na presença de radiação espacial. Também foi descrito o Sistema de Determinação de Atitude com Tolerância a Falhas (SDATF), uma proposta para operação do circuito em alta disponibilidade na baixa órbita da Terra.

No capítulo que se segue será apresentada a abordagem proposta nesse trabalho para validação da solução tolerante a falhas. Conforme será abordado, o uso de modelagem e simulação com o propósito de validação, apesar de trazer flexibilidade a esse processo, também apresenta o desafio de capturar as características essenciais ao comportamento do sistema de forma estruturada e procedural.

4.1 Métodos de Validação de Sistemas com Tolerância a Falhas do Tipo Single Event Upset (SEU)

Existem diferentes métodos de validar uma solução tolerante a falhas do tipo Single Event Upset (SEU), sendo que o intuito de tais métodos é reproduzir da forma mais fidedigna possível o ambiente ao qual o circuito estará submetido durante sua operação ou então os efeitos provenientes do fenômeno ocasionado pelo ambiente sobre os elementos de memória do circuito integrado em uso no sistema sob validação (KALASHNIKOV, 2011; MARSHALL et al., 2005).

A maneira mais realista de validação do circuito consiste em submetê-lo a uma fonte de radiação real, procedimento que exige instrumentação adequada se realizado em laboratório na Terra, ou então submeter o sistema em condições ambientais por longa duração, onde o sistema em funcionamento ficará exposto aos efeitos das partículas de alta energia, responsáveis pelo surgimento eventual dos SEUs (LIM; MARTIN; HUGHES, 1986). O primeiro método possui como vantagem maior precisão na avaliação do desempenho

do circuito sob os efeitos das partículas radioativas em laboratório, mas a maior desvantagem está relacionada ao custo envolvido no experimento - o ambiente e as pessoas envolvidas devem estar capacitados para lidar com a radiação, bem como o material deve ser descartado após os testes (MUKHERJEE, 2011, pp. 62-66).

Outro método de validação consiste na incidência de feixes de laser pulsados sobre o circuito a ser testado (BALASUBRAMANIAN et al., 2008). Esse procedimento oferece a vantagem maior capacidade de precisão referente ao tempo e ao local de incidência do feixe de partículas sobre células de memória específicas do sistema, além de permitir a repetição do experimento antes de descartá-lo (JR; MORENO; FALQUEZ, 1997). Apesar dessas vantagens, assim como a exposição de partes do sistema a uma fonte real de radiação, o uso de feixes de laser pulsados envolve alto custo com instrumentação e operação.

Além dos métodos apresentados, existe a possibilidade de se emular a ocorrência de falhas, procedimento denominado injeção de falhas (TORRES, 2013). A injeção de falhas pode ser feita por meio de hardware (TORRES, 2013; SCHIRMEIER; RADEMACHER; SPINCZYK, 2014) ou de software (COTRONEO et al., 2012; ZHANG; LIU; ZHOU, 2011). A injeção de falhas por meio de hardware é feita com o uso de interrupções externas ou acesso direto a memória. Já a injeção de falhas por meio de software é feita por meio de modificações em variáveis e registradores no software, em tempo de compilação ou em tempo de execução. Há também a possibilidade de se utilizar alguma linguagem de descrição de hardware para emular o comportamento da falha no sistema (SHOKROLAH-SHIRAZI; MIREMADI, 2008). Nas abordagens de injeção por hardware é necessária instrumentação que permita a geração de falhas e a depuração das mesmas, ao passo que na injeção por software nem sempre se tem acesso a todas as regiões sensíveis a erros do sistema (HSUEH; TSAI; IYER, 1997).

4.2 Validação por Modelagem e Simulação

Em qualquer um dos métodos citados para a validação do sistema tolerante a falhas do tipo SEU é necessária a implementação do sistema, o que envolve custos e tempo antes de avaliar se a solução proposta atende ou não as especificações que variam bastante em função da aplicação alvo e do ambiente no qual funcionará o sistema. Nesse cenário, uma abordagem baseada em modelagem e simulação se apresenta como alternativa de validação mais profícua (CALANDRA, 1991; PEREZ-CELIS et al., 2014).

Na validação por modelagem, deve-se expressar o sistema tolerante a falhas por meio de um formalismo as características de comportamento do sistema de forma não ambígua. Trata-se de uma abstração que se propõe a capturar os aspectos relevantes para a validação funcional e temporal sem necessariamente lidar com detalhes de implementação. Uma vez definido o modelo representativo do sistema, em um passo subsequente é realizada a simulação de seu comportamento em um cenário sem e com a injeção de falhas (PEREZ-

CELIS et al., 2014).

Muito se vê na literatura a respeito de validação de sistemas com tolerância a falhas por meio de ensaios com protótipos. Apesar da vantagem mencionada em se validar um sistema por meio de modelagem e simulação, não há ampla divulgação de procedimentos sistemáticos voltados a esse fim.

Poucas iniciativas com esse propósito são encontradas na literatura, como o trabalho de Perez-Celis (PEREZ-CELIS et al., 2014), que propõe uma metodologia para simulação de sistemas espaciais com tolerância a falhas baseando-se em um modelo matemático de probabilidades. Na área de sistemas distribuídos há o trabalho de Mehresh e Upadhyaya (MEHRESH; UPADHYAYA; KWIAT, 2010), que propuseram um procedimento de simulação baseado em três premissas - decomposição modular, composição independente desses módulos e parametrização. Além desses trabalhos, é possível encontrar aplicações de injeção de falhas baseadas em simulação por software (KANAWATI; KANAWATI; ABRAHAM, 1995).

Nas seções seguintes será apresentado um procedimento para modelagem do SDATF como exemplo de um processo que pode ser adaptado a outros sistemas com características semelhantes.

4.3 Modelagem baseada em Máquina de Estados Finitos

Harel (HAREL D., 1985) faz uma distinção entre *sistemas transformacionais* e *sistemas reativos*. Segundo ele, os sistemas reativos são orientados por eventos e continuamente respondem à ocorrência dos mesmos, sejam eles eventos internos ou externos ao sistema, ao passo que os sistemas transformacionais consistem basicamente em um mapeamento direto entre entradas e saídas, assumindo geralmente o formato de uma função.

Sob esse ponto de vista, um sistema tolerante a falhas pode ser classificado como um sistema reativo, visto que seu comportamento está condicionado a determinados estímulos e à ocorrência de eventos específicos, tais como interrupções e transferência de dados.

Um formalismo bastante utilizado na modelagem de sistemas reativos é a chamada Máquina de Estados Finitos (FSM – *Finite State Machine*) (GILL et al., 1962, pp. 9-14). Uma Máquina de Estados Finitos consiste em um grafo, onde cada vértice representa um estado do sistema e cada aresta representa uma transição entre os estados, conforme mostrado no exemplo da Figura 16.

Nesse exemplo, a transição é caracterizada por uma entrada (ou evento) e uma saída. A transição do estado x_0 para o estado x_1, por exemplo, ocorre a partir da entrada b e resulta na saída 1.

Uma Máquina de Estados Finitos G pode ser definida pela sêxtupla

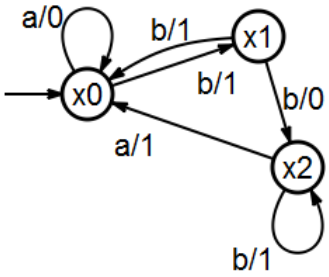


Figura 16 – Exemplo de uma Máquina de Estados Finitos.

$$G = \{X, Z, E, f, \Gamma, x_0\} \tag{6}$$

onde X é o conjunto de estados do sistema, Z é o conjunto de saídas possíveis, E é o conjunto de entradas/eventos, $f : X \times E \rightarrow X \times Z$ é a função de transição entre os estados, $\Gamma : X \rightarrow 2^E$ é o conjunto das entradas/eventos factíveis a um dado estado, e x_0 representa o estado inicial.

As máquinas de estados finitos consistem numa poderosa ferramenta para se representar formalmente o comportamento de um sistema, usando os conceitos básicos relacionados a estados e transições. No entanto, à medida que o sistema a ser modelado se torna mais complexo, a sintaxe provida pelas FSM's se torna ineficiente. Em situações nas quais há configurações distintas no sistema, a modelagem por máquina de estados finitos sofre com o efeito denominado explosão de estados (PARNAS, 1969, p. 380). Nesse caso a representação bidimensional dessa ferramenta não é a mais adequada. Além disso, muitos sistemas apresentam características de assincronismo e paralelismo, o que não é explicitamente modelado por uma máquina de estados finitos clássica.

Outro formalismo, também baseado na ideia de estados e transições, são os chamados *Statecharts* (HAREL, 1987). Trata-se por um formalismo proposto por David Harel ao final da década de 80, que tem por objetivo a modelagem de sistemas complexos de forma simplificada, através de uma especificação hierárquica. Os Statecharts possuem uma sintaxe que lida explicitamente com os problemas mencionados.

4.4 Modelagem usando Statecharts

Os Statecharts constituem um formalismo derivado da noção básica de estados e transições/eventos, propostos na abordagem por máquina de estados finitos, acrescido de características que tornam essa linguagem eficiente na modelagem de sistemas complexos (HAREL, 1987).

Harel definiu a sintaxe utilizada nos statecharts em seu *paper* (HAREL et al., 1987), e resumiu a ideia dessa ferramenta no uso de diagrama de estados, dotado dos seguintes

recursos – profundidade, ortogonalidade e comunicação em *broadcast*. Cada uma dessas características será explicitada a seguir.

O conceito de profundidade está relacionado ao agrupamento de estados de forma hierárquica. Seja o modelo mostrado na Figura 17.

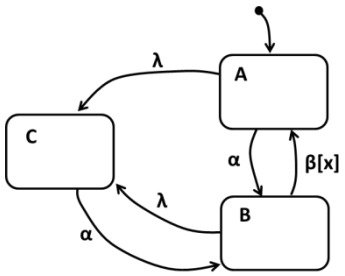


Figura 17 – Exemplo de statechart.

Esse modelo representa três estados, A, B e C, sujeitos à ocorrência dos eventos α, β e λ, onde A é o estado inicial. A ocorrência representada por $\beta[x]$ indica que a transição do estado B para o estado A acontecerá na ocorrência do evento β se a condição x for verdadeira. Deve ser observado que estando o sistema em A ou em B, ele deve assumir o estado C na ocorrência do evento λ. Em outras palavras, é como se os estados A e B fizessem parte de uma situação na qual o sistema está sujeito à transição para o estado C caso λ ocorra. Dessa forma, o sistema pode ser representado de forma equivalente pelo statechart mostrado na Figura 18.

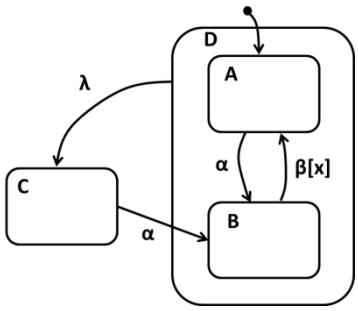


Figura 18 – Exemplo de uso de profundidade na modelagem.

Nessa representação, os estados A e B foram agrupados no estado D, conceito denominado profundidade (HAREL et al., 1987). Esse simples conceito permite a modelagem de um sistema complexo de forma hierárquica. Um modelo inicial, como mostrado na Figura 19, poderia ser proposto levando-se em conta apenas os estados C e D. Uma vez especificado o comportamento do sistema nesse nível, o estado D poderia ser detalhado, dando origem ao modelo da Figura 18.

Outro conceito bastante utilizado na modelagem por statecharts é o de ortogonalidade. Na Figura 18, por exemplo, diz-se que os estados A e B são uma composição XOR do

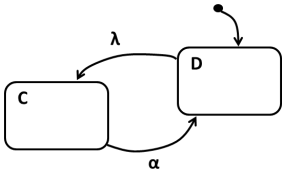


Figura 19 – Exemplo de modelagem hierárquica.

estado D. Isso significa que, estando o sistema no estado D, ele só pode estar em A ou em B – nunca nos dois ao mesmo tempo e, estando em D, necessariamente deve estar em um dos dois (A ou B). O conceito de ortogonalidade consiste também na composição AND de estados, como mostrado na Figura 20.

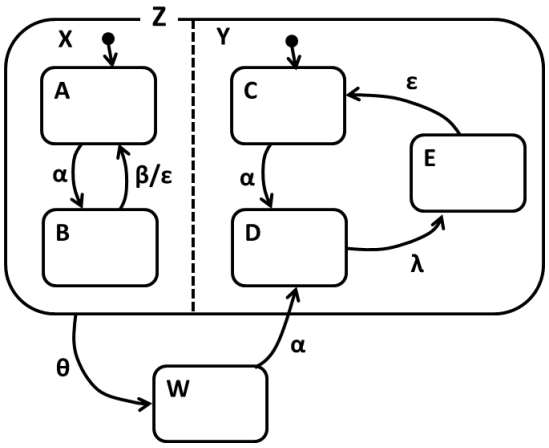


Figura 20 – Ortogonalidade em statechart.

Nessa figura o estado Z é dividido em dois estados ortogonais, X e Y, que atuam em paralelo. Por se tratar de uma composição AND, estando o sistema no estado Z, ele sempre estará numa combinação entre os conjuntos $\{A, B\}$ e $\{C, D, E\}$.

Essa característica do formalismo é bastante útil na modelagem de sistemas reais. Nesse caso o sistema poderia ser dividido em componentes funcionais e cada componente pode ser modelado separadamente. O modelo resultante do sistema será definido pela composição ortogonal dos componentes. A Figura 21, retirada de (HAREL, 1987), mostra um exemplo de modelo em alto nível para um sistema aviônico usando o conceito de ortogonalidade.

Além da característica de ortogonalidade, o statechart da Figura 20 também exemplifica o conceito de comunicação em broadcast. Suponha que em um dado instante o sistema esteja no estado Z, especificamente nos estados B e E. Em seguida, suponha a ocorrência do evento β. Deve ser observada a notação β/ϵ, que indica que o disparo do evento ϵ é consequência da ocorrência de β.

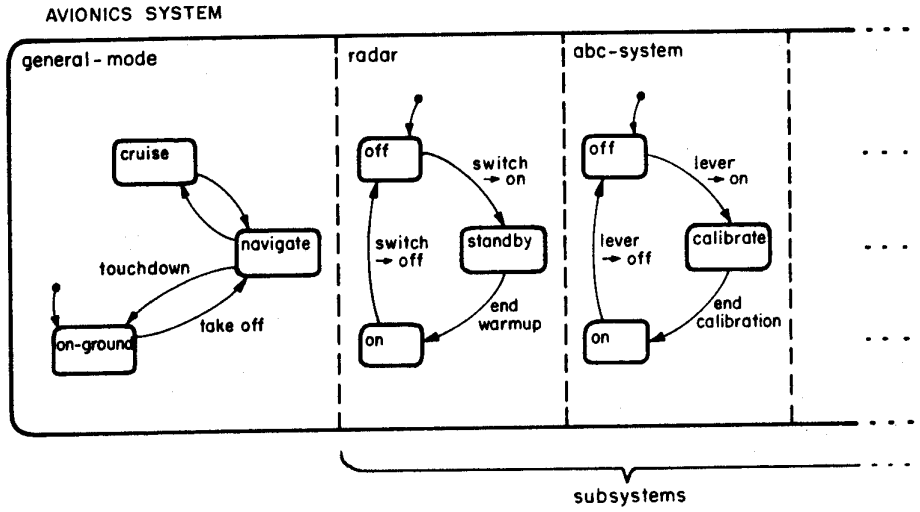


Figura 21 – Exemplo de modelo em alto nível de um sistema aviônico.

Fonte: (HAREL, 1987)

De acordo com a semântica do statechart (HAREL et al., 1987), essa situação deve ser interpretada como ϵ ocorrendo no instante imediatamente seguinte ao disparo de β. Assim sendo, se o sistema estiver nos estados B e E, ele sofrerá transição para os estados A e C quando β ocorrer. A essa influência que um estado faz no estado ortogonal a ele se dá o nome de *comunicação em broadcast*. Essa característica permite que a modelagem de um sistema com base na divisão de componentes represente a interação entre os mesmos.

O modelo apresentado na Figura 20 exemplifica as três características principais dos statecharts – a profundidade é exemplificada pela transição que ocorre com o evento θ, pois é válida para todos os estados interiores a Z; a ortogonalidade é representada pelos estados X e Y; e a comunicação em broadcast é representada pela transição β/ϵ. Essas características permitem um formalismo bastante poderoso, capaz de fornecer um modelo simples, não ambíguo e de fácil compreensão para sistemas complexos. A Figura 22 mostra um modelo equivalente ao modelo da Figura 20, porém sem o uso das características aqui mencionadas.

O modelo mostrado na Figura 22, além de ser mais difícil de ser desenvolvido, também é de mais difícil compreensão, o que facilita a ocorrência de erros no desenvolvimento do sistema real, caso tal modelo seja utilizado como especificação. Além disso, o modelo difuso da Figura 22 apresenta maior dificuldade para realização de alterações no caso de uma mudança nos requisitos do sistema.

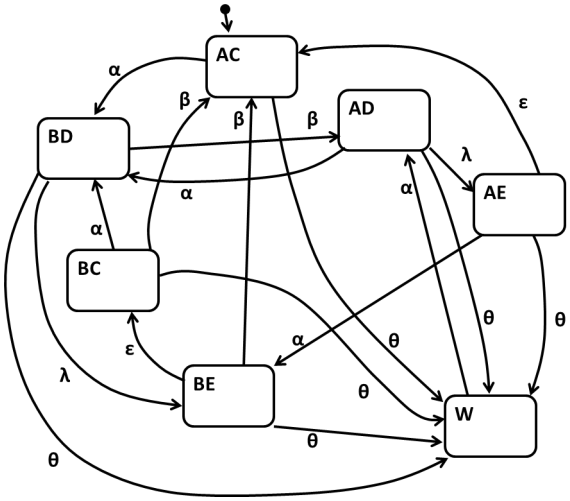


Figura 22 – Modelagem sem o uso de profundidade, ortogonalidade e comunicação em broadcast.

4.4.1 Alguns aspectos relevantes da sintaxe em Statechart

O formalismo proposto por D. Harel evoluiu ao longo dos anos, tornando-se uma linguagem de descrição de sistemas bastante completa, dotada de diversos recursos úteis à descrição de sistemas com comportamento complexo.

A Figura 23 apresenta um exemplo de statechart com alguns recursos usados na descrição do comportamento de um sistema genérico.

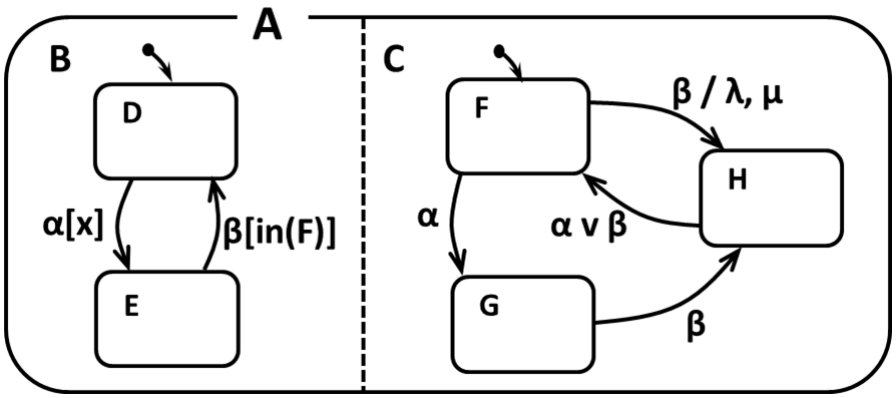


Figura 23 – Statechart como exemplo de alguns recursos da modelagem.

Nesse exemplo, um estado A é dividido em dois comportamentos concorrentes, representados pelos estados B e C em composição ortogonal. O comportamento em B é dividido em dois estados, D e E, ao passo que o comportamento em C é dividido nos estados F, G e H.

O sistema no estado A está sujeito a dois eventos, α e β, e possui duas saídas booleanas, λ e μ. A partir deste exemplo alguns recursos do statechart podem ser estabelecidos:

- Transição condicional → A transição condicional já foi abordada no exemplo da Figura 17. Na Figura 23 ela está presente nas transições entre os estados D e E. Na transição de D para E, por exemplo, temos $\alpha[x]$. Isso significa que, estando o sistema no estado D, caso ocorra o evento α, o sistema assumirá o estado E somente se a condição x for verdadeira;

- Função "in" → A função "in" verifica se determinado estado está ativo dentro de algum dos estados em composição ortogonal. Esse recurso é usado na condição da transição de E para D, dada por $\beta[in(F)]$. Caso o sistema esteja no estado E e o evento β ocorra, ocorrerá a transição para D apenas se, no estado C em composição ortogonal, o estado F estiver ativo;

- Operações lógicas → Operações lógicas também podem ser utilizadas entre condições e eventos. Na transição do estado H para o estado F, por exemplo, é utilizada a operação lógica OU. Essa operação é representada pelo símbolo $\vee$. Estando o sistema em H, ele assumirá o estado F se ocorrer o evento α ou se ocorrer o evento β;

- Saídas como consequência de eventos → A ocorrência de eventos podem disparar não apenas transições no modelo, como também podem ativar saídas. Quando o sistema está no estado F, por exemplo, a transição para o estado H será disparada pelo evento β. Como consequência desse evento, as saídas λ e μ serão ativadas.

4.5 Modelo em Statechart para o comportamento do Sistema de Determinação de Atitude com Tolerância a Falhas (SDATF)

Além das máquinas de estados finitos, existem diversos formalismos que podem ser utilizados para descrever o comportamento de um sistema tolerante a falhas do tipo SEU. SysML (FRIEDENTHAL; MOORE; STEINER, 2014), AADL (FEILER; GLUCH; HUDAK, 2006) e Redes de Petri (MURATA, 1989) são alguns exemplos. Para o procedimento proposto neste trabalho, o Statechart foi o formalismo escolhido devido à sua simplicidade e a adequação das três principais características (profundidade, ortogonalidade e comunicação em *broadcast*) à logica de funcionamento do sistema tolerante a falhas.

Conforme mencionado, o sistema tolerante a falhas pode ser classificado como um sistema reativo. Além disso, o SDATF apresenta características comuns a sistemas complexos, tais como a divisão em componentes, o paralelismo e o comportamento assíncrono

(as rotinas são orientadas a interrupções, que ocorrem de forma independente entre os microcontroladores). Sendo assim, é pertinente o uso do formalismo proposto nos Statecharts para modelar o comportamento desse sistema.

Numa abordagem inicial, pode-se pensar no sistema como sendo a execução em paralelo dos três microcontroladores, os quais serão denominados MC_i, MC_j e MC_k. Assim, são definidos três estados ortogonais como na Figura 24.

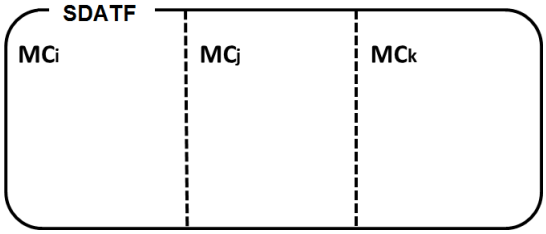


Figura 24 – SDATF como composição ortogonal dos três microcontroladores.

Definindo-se dessa forma, o próximo passo consiste em modelar o comportamento de um microcontrolador. Como os três microcontroladores executam o mesmo software, espera-se que o modelo interno dos estados MC_i, MC_j e MC_k sejam análogos, ou seja, que cada microcontrolador seja um instância do mesmo modelo em statechart. Assim sendo, será detalhado o estado MC_i e seu comportamento deverá ser estendido aos demais. O modelo proposto é mostrado na Figura 25.

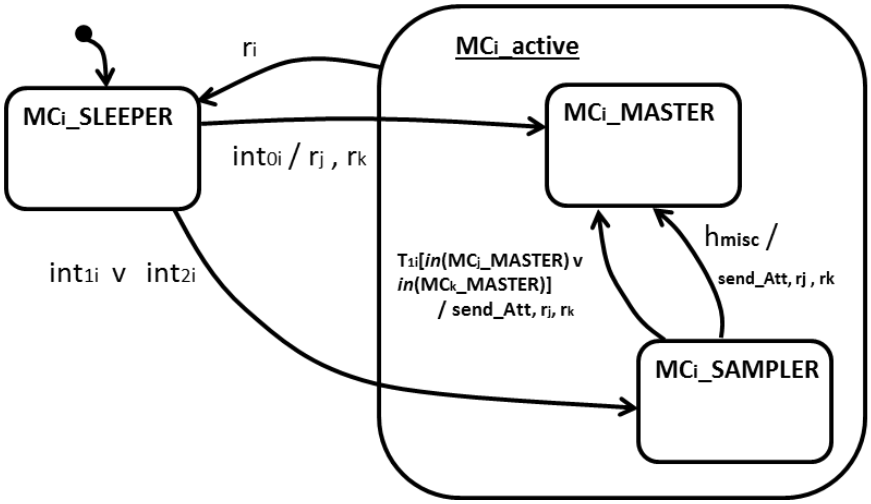


Figura 25 – Statechart que representa o comportamento de um microcontrolador do SDATF.

No modelo da Figura 25, o microcontrolador deve assumir um dos três estados (papéis descritos no Capítulo 3) – SLEEPER, MASTER ou SAMPLER ($MC_i\_SLEEPER$, $MC_i\_SAMPLER$ e MC_i_MASTER, respectivamente para o microcontrolador i), sendo que no estado inicial ele assume o papel de SLEEPER. Uma vez no estado SLEEPER,

o microcontrolador pode sofrer três interrupções, denominadas int_0, int_1 e int_2. Tais interrupções são explicadas a seguir:

- int_0 - Interrupção através da qual o computador de bordo do satélite sinaliza que o microcontrolador deve assumir o papel de MASTER;
- int_1 - Interrupção através da qual o microcontrolador "à esquerda"[1] sinaliza que o microcontrolador deve assumir o papel de SAMPLER;
- int_2 - Interrupção através da qual o microcontrolador "à direita" sinaliza que o microcontrolador deve assumir o papel de SAMPLER.

Na notação da Figura 25, deve ser entendido que int_{0i} indica que o microcontrolador i sofreu a interrupção int_0. Analogamente, int_{1i} e int_{2i} indicam que o microcontrolador i sofreu as interrupções int_1 e int_2, respectivamente, bem como int_{0j} indica que o microcontrolador j sofreu a interrupção int_0, e assim por diante.

Além das interrupções, o microcontrolador pode receber um sinal de *reset*, voltando ao estado inicial. Considerando a mesma notação, r_i indica que o microcontrolador i recebeu o sinal de reset, e r_j e r_k são os eventos nos quais os microcontroladores j e k, respectivamente, recebem sinais de reset e voltam ao estado inicial.

Enquanto que no estado SLEEPER o microcontrolador está em modo de baixo consumo, ao sofrer uma das interrupções mencionadas ele passa a um estado ativo, sendo MASTER ou SAMPLER. A qualquer instante o MC pode receber um sinal de reset (r_i na Figura), visto que tal sinal é um evento externo, proveniente da operação independente dos demais microcontroladores. Se isso ocorrer ele deve voltar para a condição de SLEEPER. Da mesma forma, microcontrolador i pode causar eventos de reset nos demais microcontroladores (r_j e r_k).

No estado SLEEPER, os seguintes eventos podem ocorrer (para eventos não explicitados, considera-se que o sistema permanece no mesmo estado):

- $int_{0i}/r_j, r_k$ – Indica a ocorrência da interrupção int_0 no microcontrolador i. Nesse caso o MC_i passa para o estado MASTER e envia sinal de reset para os demais microcontroladores (r_j, r_k);
- $int_{1i} \vee int_{2i}$ – Indica que o microcontrolador i sofreu a interrupção int_1 (quando ele é "acordado" pelo microcontrolador "à esquerda", ou seja, o microcontrolador k) ou (a operação lógica *OR* é indicada pelo símbolo "$\vee$") sofreu a interrupção int_2 (quando ele é "acordado"pelo microcontrolador "à direita", ou seja, o microcontrolador j). Nesse caso o MC_i assume o estado SAMPLER.

[1] As expressões "à esquerda" e "à direita" serão utilizadas ao longo desse texto para referir aos demais microcontroladores. Para o MC_i, os microcontroladores MC_k e MC_j estão "à esquerda" e "à direita", respectivamente. Para MC_j, temos os microcontroladores MC_i e MC_k, e para o MC_k, os microcontroladores MC_j e MC_i, respectivamente.

Uma vez no papel de SAMPLER, há situações nas quais o microcontrolador deve enviar um sinal de reset aos demais MCs e assumir o estado MASTER. Isso deve ocorrer nos seguintes eventos:

- $T_{1i}[in(MC_j\_MASTER) \vee in(MC_k\_MASTER)]/send\_Att, r_j, r_k$ - T_{1i} indica o estouro do *timer* 1 no microcontrolador i. O significado desse timer varia conforme o estado assumido pelo MC. Estando no papel de SAMPLER, esse timer indica o tempo máximo durante o qual o microcontrolador deve aguardar pelo envio do sinal de *heartbeat* por parte do MASTER. O MASTER, nessa situação pode ser assumido pelo MC_j ou pelo MC_k. O termo "*in*", na sintaxe de statechart, representa a condição de um estado ortogonal estar ou não em um determinado subestado (HAREL et al., 1987). Nesse caso a condição $[in(MC_j\_MASTER) \vee in(MC_k\_MASTER)]$ será verdadeira se o microcontrolador j ou o microcontrolador k estiverem no estado MASTER. Dessa forma, quando o microcontrolador i está no estado SAMPLER e o timer 1 estoura (T_{1i}), se o MC_j ou o MC_k estiverem no papel de MASTER, o MC_i sofrerá transição do estado SAMPLER para o estado MASTER, enviando sinal de reset aos demais microcontroladores e enviando a última atitude válida ao computador de bordo do satélite ($send\_Att, r_j, r_k$);

- $h_{misc}/send\_Att, r_j, r_k$ – No estado SAMPLER, o canal UART[2] do microcontrolador está habilitado. Na ocorrência de uma interrupção que indique um recebimento por esse canal ($UART_{RX}$), ele verifica se a informação recebida equivale ao valor corrente do *heartbeat*. Caso o valor recebido seja diferente (*heartbeat miscompare*), ele assume o papel de MASTER e envia um sinal de reset aos demais microcontroladores e envia a última atitude válida ao computador de bordo do satélite ($send\_Att, r_j, r_k$).

A ocorrência de tais eventos ficará mais clara na medida em que o modelo for detalhado. Se o microcontrolador estiver no estado MASTER, ele só irá deixar esse estado ao receber o sinal de reset (r_i), voltando à condição de SLEEPER.

Voltando ao escopo do SDATF, pode-se associar uma instância do modelo da Figura 25 a cada microcontrolador representado na Figura 24. O resultado é mostrado na Figura 26.

Resta agora detalhar o comportamento de cada microcontrolador em cada um dos papéis. No estado SLEEPER o MC está apenas aguardando por interrupções, portanto serão detalhados os estados ativos, MASTER e SAMPLER.

A Figura 27 mostra o modelo proposto para o comportamento do microcontrolador quando assumido o papel de MASTER. Nessa Figura considerou-se o MC_i como referência. Os modelos para MC_j e MC_k são análogos.

Quando no estado MASTER, o microcontrolador possui duas tarefas principais – sua operação interna (estado "internal_operation"), na qual ele executa a rotina de sincronismo em conjunto com o SAMPLER, e a sincronização com o computador de bordo

[2] Universal Asynchronous Receiver/Transmitter

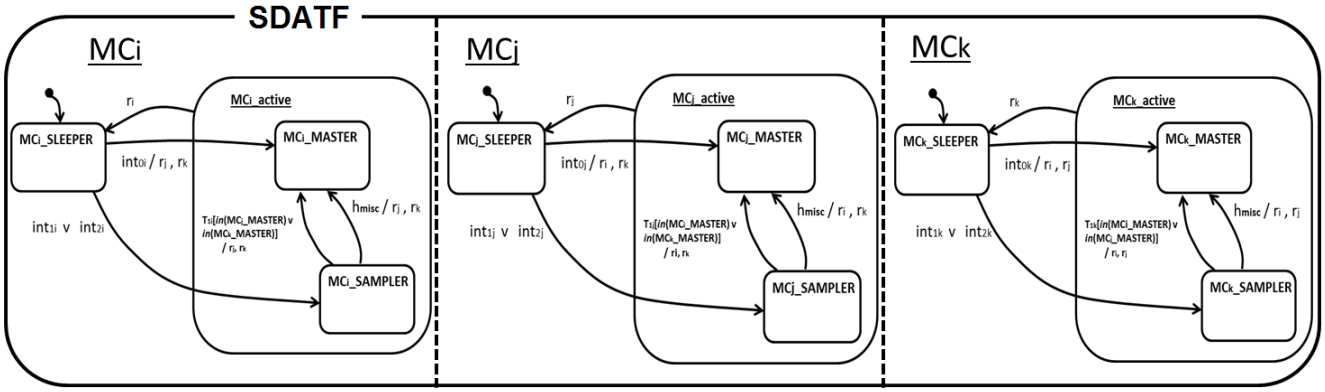


Figura 26 – Detalhamento do modelo com os microcontroladores em composição ortogonal.

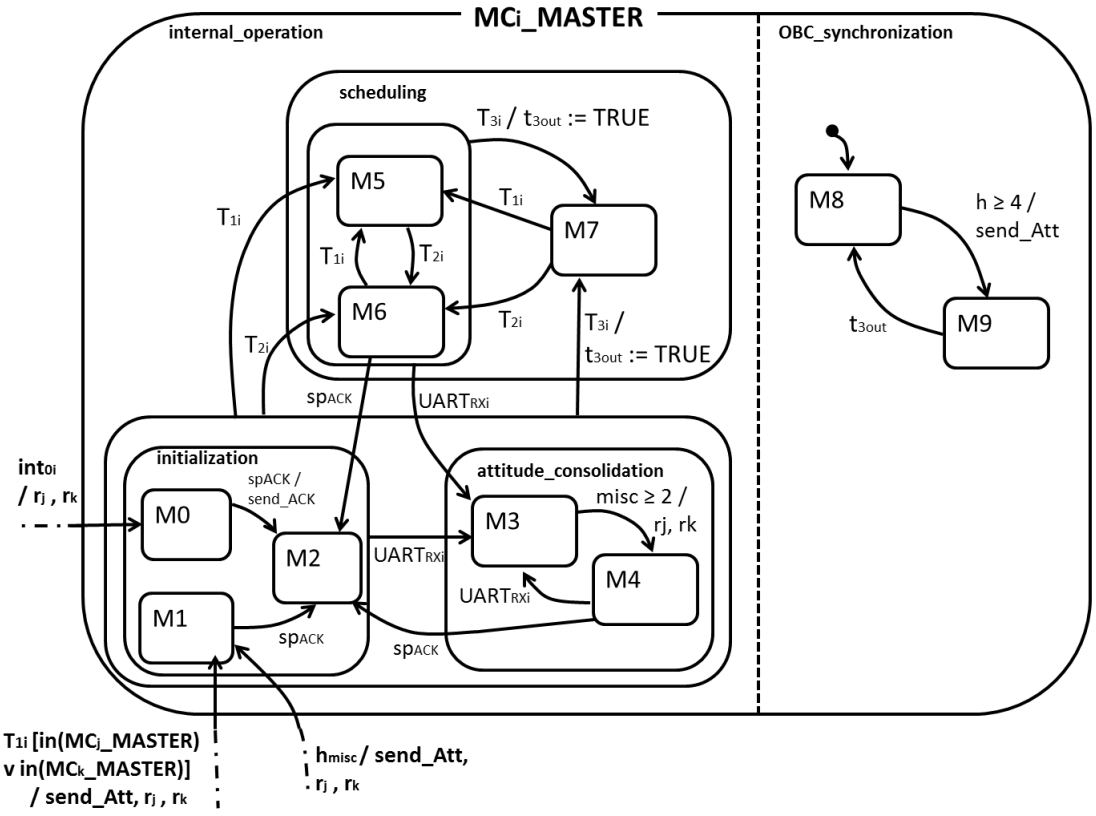


Figura 27 – Modelo para o comportamento do MASTER.

(estado "OBC_synchronization"), na qual há o envio da atitude consolidada. Essas duas tarefas foram modeladas como estados ortogonais.

Os eventos que determinam a entrada no estado MASTER são:

- int_{0i} – Interrupção através da qual o microcontrolador é "acordado"pelo OBC para assumir o papel de MASTER. Nesse caso o MC passa para o estado M_0. Nesse

estado, o MC envia um sinal de reset aos demais microcontroladores (r_j, r_k), é inicializado como MASTER, calcula a atitude e armazena como última atitude válida, e em seguida inicializa um dos demais microcontroladores para assumir o papel de SAMPLER. O MC armazena em uma variável interna a informação de qual microcontrolador (MC_j ou MC_k) deverá ser "acordado", o que determina o disparo do evento int_{1j} ou int_{2k}. No caso de falha em estabelecer uma rotina de sincronismo com o SAMPLER, a variável que indica o MC a sofrer interrupção é modificada. É importante ressaltar que o disparo de int_{1j} ou int_{2k} está implícito nos estados M_0 e M_1, uma vez que faz parte da rotina executada quando nesses estados;

- $T_{1i}[in(MC_j\_MASTER) \vee in(MC_k\_MASTER)]/send\_Att, r_j, r_k$ – Representa o estouro do timer 1 quando o microcontrolador i está no papel de SAMPLER. Trata-se do mesmo evento abordado anteriormente na transição de SAMPLER para MASTER. Tal evento, portanto, indica que o microcontrolador estava no papel de SAMPLER anteriormente ao estouro do timer 1, o que gerou a transição para o estado M_1. Nesse estado, o MC envia um sinal de reset aos demais microcontroladores e a última atitude válida ao OBC $(send\_Att, r_j, r_k)$, assumindo o papel de MASTER. Em seguida, ele inicializa um dos demais microcontroladores para assumir o papel de SAMPLER. Assim como no estado M_0, a rotina executada em M_1 pode disparar os eventos int_{1j} ou int_{2k} dependendo de qual MC deverá ser instanciado como SAMPLER;

- $h_{misc}/send\_Att, r_j, r_k$ – Indica que o microcontrolador estava inicialmente no estado SAMPLER e detectou um *heartbeat miscompare*. Esse evento também foi discutido anteriormente. Nesse caso, ele assume o papel de MASTER, em transição para o estado M_1, enviando sinal de reset aos demais microcontroladores e a última atitude válida ao OBC $(send\_Att, r_j, r_k)$.

Quando o microcontrolador MC_i é configurado no modo MASTER, as interrupções int_{0i}, int_{1i} e int_{2i} são desabilitadas, ou seja, ele não pode ser sinalizado pelo computador de bordo para assumir o papel de MASTER (int_{0i}) ou pelos demais microcontroladores para assumir o papel de SAMPLER (int_{1i} e int_{2i}). No escopo desse modelo, portanto, se alguma dessas interrupções ocorrerem enquanto o microcontrolador MC_i está no papel de MASTER, nada ocorre e ele permanece no mesmo estado.

As seguintes interrupções são habilitadas, associadas a eventos:

- T_{1i} – Estouro do timer 1, indicando que o MC deve enviar o *heartbeat* ao SAMPLER;

- T_{2i} – Estouro do timer 2, indicando término do tempo máximo de espera pelo envio da atitude calculada pelo SAMPLER;

- T_{3i} – Estouro do timer 3, indicando término da janela de tempo de 1 segundo na qual a atitude consolidada deve ser enviada ao OBC;
- $UART_{RXi}$ – Interrupção do canal de comunicação serial, indicando recebimento de informação.

Além das interrupções, alguns eventos são definidos quando algumas variáveis internas assumem certos valores. As seguintes variáveis são consideradas:

- h_{misc} – Conforme já mencionado, trata-se de uma variável booleana que indica se houve ou não *heartbeat miscompare*;
- h – Inteiro que armazena o valor do *heartbeat* a ser enviado ao SAMPLER;
- $misc$ – Inteiro que armazena o número de vezes em que a atitude recebida pelo SAMPLER foi considerada inválida;
- t_{3out} – Variável booleana que indica se houve ou não o estouro do timer 3;
- sp_{ACK} – Variável booleana que indica a resposta do SAMPLER ao ser inicializado;

No modelo proposto, duas variáveis de saída são consideradas:

- *send_ACK* – Variável que indica envio de *Acknowledgement* (ACK) para o computador de bordo, indicando que o microcontrolador foi inicializado como MASTER e que a rotina de sincronismo com o SAMPLER foi estabelecida;
- *send_Att* – Variável que sinaliza envio da atitude consolidada ao computador de bordo.

A seguir, cada estado é explicado.

Inicialização (estado "initialization")

M_0 Conforme mencionado, é o estado de inicialização no qual o microcontrolador é "acordado" pelo computador de bordo do satélite. Nesse estado, o MC_i é inicializado como MASTER, calcula a atitude e armazena como última atitude válida, envia um sinal de reset aos demais microcontroladores e em seguida inicializa um deles para assumir o papel de SAMPLER. Ao estabelecer a rotina de sincronismo com o SAMPLER, o microcontrolador deve enviar um sinal de *Acknowledgement* (ACK) ao computador de bordo do satélite. A escolha de qual microcontrolador deve ser inicializado como SAMPLER é arbitrária. O importante é que, caso o microcontrolador a ser inicializado não apresente resposta a tempo, o MASTER deve inicializar o outro microcontrolador, alternando a vez de cada um;

M_1 Semelhante ao estado M_0, porém esse estado é assumido quando o MC_i assume o papel de MASTER estando antes no papel de SAMPLER. Nesse estado, o MC_i envia a última atitude válida armazenada para o computador de bordo e envia um sinal de reset aos demais microcontroladores, assumindo o papel de MASTER. Em seguida, ele inicializa um dos demais microcontroladores para assumir o papel de SAMPLER;

M_2 É o estado assumido pelo MC após estabelecer uma rotina de sincronismo com o SAMPLER (quando a variável booleana "sp_{ACK}"é verdadeira). Na transição a partir de M_0, é sinalizado o envio do sinal de ACK para o computador de bordo ($send\_ACK$). No estado M_2 o microcontrolador aguarda pelo disparo de alguma interrupção.

Consolidação da atitude (estado "attitude_consolidation")

Ao sofrer a interrupção $UART_{RX}$ o microcontrolador passa para um estado de consolidação da atitude.

M_3 Nesse estado, o MC lê a informação recebida pelo canal UART. Tratando-se da atitude, ele valida essa atitude em comparação com a última atitude válida armazenada. Se a atitude recebida for considerada válida, ele armazena essa informação como a última atitude válida, caso contrário incrementa o contador "$misc$". Além disso, o timer 2 é inicializado;

M_4 O MC assume esse estado quando $misc \geq 2$. Nesse caso, o microcontrolador envia um sinal de reset aos demais microcontroladores e inicializa o MC que estava no papel de SLEEPER para assumir o papel de SAMPLER.

Escalonamento (estado "scheduling")

Os estados relacionados ao escalonamento são acionados quando algum timer estoura.

M_5 O microcontrolador entra nesse estado quando o timer 1 estoura. Nesse caso, o MC envia o *heartbeat* ao SAMPLER e incrementa o valor de "h". Em seguida, o timer 1 é inicializado novamente;

M_6 O microcontrolador entra nesse estado quando o timer 2 estoura. Quando isso ocorre, significa que a atitude por parte do SAMPLER não foi recebida a tempo. Nesse caso, o MC envia um sinal de reset aos demais microcontroladores e a última atitude

válida ao OBC[3]. Em seguida, ele inicializa o MC que estava no papel de SLEEPER para assumir o papel de SAMPLER;

M_7 O microcontrolador entra nesse estado quando o timer 3 estoura. Isso significa o fim de um frame de 1 segundo relativo ao envio da atitude consolidada ao OBC. Nesse caso, os timers 1, 2 e 3 são inicializados, bem como o valor do *heartbeat*. A transição para esse estado usa a variável t_{3out} para indicar aos estados de sincronização com o computador de bordo, por meio da comunicação em broadcast, que um novo frame de sincronismo deve começar (transição do estado M_9 para o estado M_8. Uma vez no estado M_8, o microcontrolador no modo MASTER está novamente sujeito a enviar a atitude consolidada ao OBC após o envio do *heartbeat* pela quarta vez).

Sincronização com o OBC (estado "OBC_synchronization")

São os estados referentes ao envio periódico da atitude consolidada ao computador de bordo do satélite.

M_8 É o estado inicial da sincronização. Trata-se de um estado que indica operação interna, na qual não há o envio da atitude consolidada;

M_9 É o estado no qual o microcontrolador deve enviar a atitude consolidada ao OBC (*send_Att*). Após o envio, ele fica aguardando o estouro do timer 3, indicando o fim do frame de 1 segundo e início de um novo frame.

Os estados abordados se referem às condições nas quais o microcontrolador assume o papel de MASTER. A Figura 28 mostra o modelo detalhado do comportamento de um microcontrolador quando assumido o papel de SAMPLER.

Esse modelo considera os mesmos eventos e variáveis utilizados no modelo do MASTER, além da variável "q". Essa variável é um inteiro que indica o número de quatérnios de atitude enviados ao MASTER.

Assim como no modelo para o MASTER, a operação do SAMPLER é dividida em duas tarefas ortogonais – operação interna (estado "internal_operation") e sincronização do frame de 1 segundo (estado "frame_synchronization"). Os eventos de entrada nos estados de SAMPLER são as interrupções int_{1i} e int_{2i} que indicam a sinalização por parte dos dois outros microcontroladores para que esse MC passe da configuração SLEEPER para SAMPLER, indo para o estado S_0. Assim como na configuração de MASTER, as interrupções int_{0i}, int_{1i} e int_{2i} são desabilitadas quando o microcontrolador é associado a essa função, no estado S_0. Na ocorrência do evento T_{1i}, indicando que o MC não recebeu

[3] As saídas $send_{Att}$, r_j e r_k estão implícitas no modelo. Elas são ativadas dentro do estado M_6, e poderiam estar explicitadas nas transições para este estado como $T_{2i}/send_{Att}, r_j, r_k$. Decidiu-se por não representar dessa forma no modelo apenas para priorizar a clareza da representação gráfica, evitando o excesso de informação na imagem.

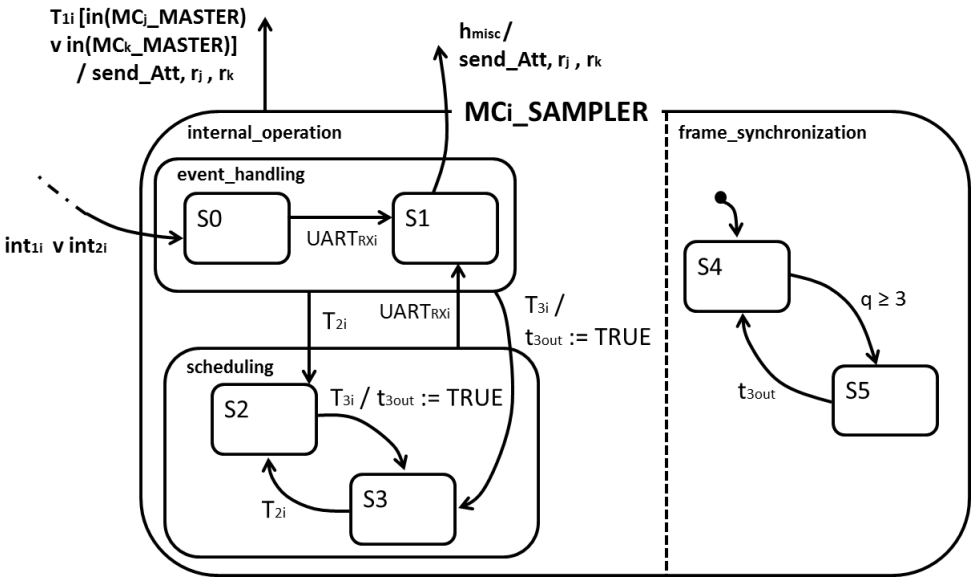


Figura 28 – Modelo para o comportamento do SAMPLER.

o *heartbeat* por parte do MASTER dentro do tempo limite, o MC envia um sinal de reset aos demais microcontroladores e assume o papel de MASTER, enviando a última atitude válida ao OBC (estado M_1).

Cada estado no modo SAMPLER é explicado a seguir.

Inicialização e Comunicação (estado "event_handling")

São os estados associados à inicialização do modo SAMPLER, e à recepção de informação por meio do canal de comunicação serial.

S_0 É o estado no qual o microcontrolador é configurado como SAMPLER. O microcontrolador atinge esse estado quando outro microcontrolador, em um dos estados M_0, M_1, M_3 ou M_5, gera a interrupção para que esse MC vá do estado SLEEPER para S_0. Assim sendo, o microcontrolador envia um sinal para confirmar que está ativo e inicializa os timers e o contador "q". Em seguida ele calcula a atitude e armazena como última atitude válida;

S_1 Esse estado é assumido na ocorrência do evento $UART_{RXi}$, que indica o recebimento de informação pelo canal UART. Nesse caso, o MC reinicia o timer 1 (tempo máximo de aguardo pelo envio do *heartbeat* por parte do MASTER) e verifica se o valor recebido equivale ao *heartbeat* esperado. Caso não seja (*heartbeat miscompare*) o evento h_{misc} é disparado e o microcontrolador assume o papel de MASTER (estado M_1), enviando sinal de reset aos demais microcontroladores e a última atitude válida ao OBC ($send\_Att, r_j, r_k$).

Escalonamento (estado "scheduling")

São os estados relacionados ao tratamento de estouro dos timers T_{2i} e T_{3i} (o estouro do timer 1 leva o MC ao estado M_1).

S_2 O microcontrolador entra nesse estado quando o timer 2 estoura. Nesse caso ele envia a última atitude válida aos demais microcontroladores pelo canal UART e incrementa o valor de "q";

S_3 O microcontrolador entra nesse estado quando o timer 3 estoura. Isso significa o fim de um frame de 1 segundo relativo ao envio da atitude consolidada ao OBC. Nesse caso, os timers 1, 2 e 3 são inicializados, bem como o valor de "q". A transição para esse estado usa a variável t_{3out} para indicar aos estados de sincronização de frame, por meio da comunicação em broadcast, que o microcontrolador deve aguardar o envio da atitude consolidada ao OBC.

Sincronização de frame (estado "frame_synchronization")

São os estados referentes ao envio periódico da atitude consolidada ao computador de bordo do satélite.

S_4 É o estado inicial da sincronização. Trata-se de um estado que indica operação interna, na qual não há o envio da atitude consolidada;

S_5 É o estado no qual o microcontrolador deve aguardar o envio a atitude consolidada ao OBC por parte do MASTER. Nesse estado o MC fica aguardando o estouro do timer 3, indicando o fim do frame de 1 segundo e início de um novo frame.

O formalismo proposto permite a modelagem de forma hierárquica do sistema tolerante a falhas. Partiu-se da proposta de três estados ortogonais (Figura 24), cada um referente a um microcontrolador e o comportamento de cada MC foi detalhado em termos dos três papéis – SLEEPER, MASTER e SAMPLER (Figura 25). Cada papel do MC foi então detalhado em mais estados.

A Figura 29 mostra o modelo equivalente ao mostrado na Figura 25, considerando os estados que detalham o comportamento do MC em cada papel.

O modelo que descreve o comportamento do Sistema de Determinação de Atitude com Tolerância a Falhas (SDATF), portanto, consiste em três instâncias ortogonais do modelo da Figura 29, tal como o mostrado na Figura 26.

No procedimento proposto nesse capítulo partiu-se de uma descrição em linguagem natural de um sistema de comportamento complexo, traduzindo a descrição desse comportamento para uma representação formal baseada em estados e transições. Essa representação torna a modelagem mais simples, uma vez que representa uma tendência "dividir para conquistar".

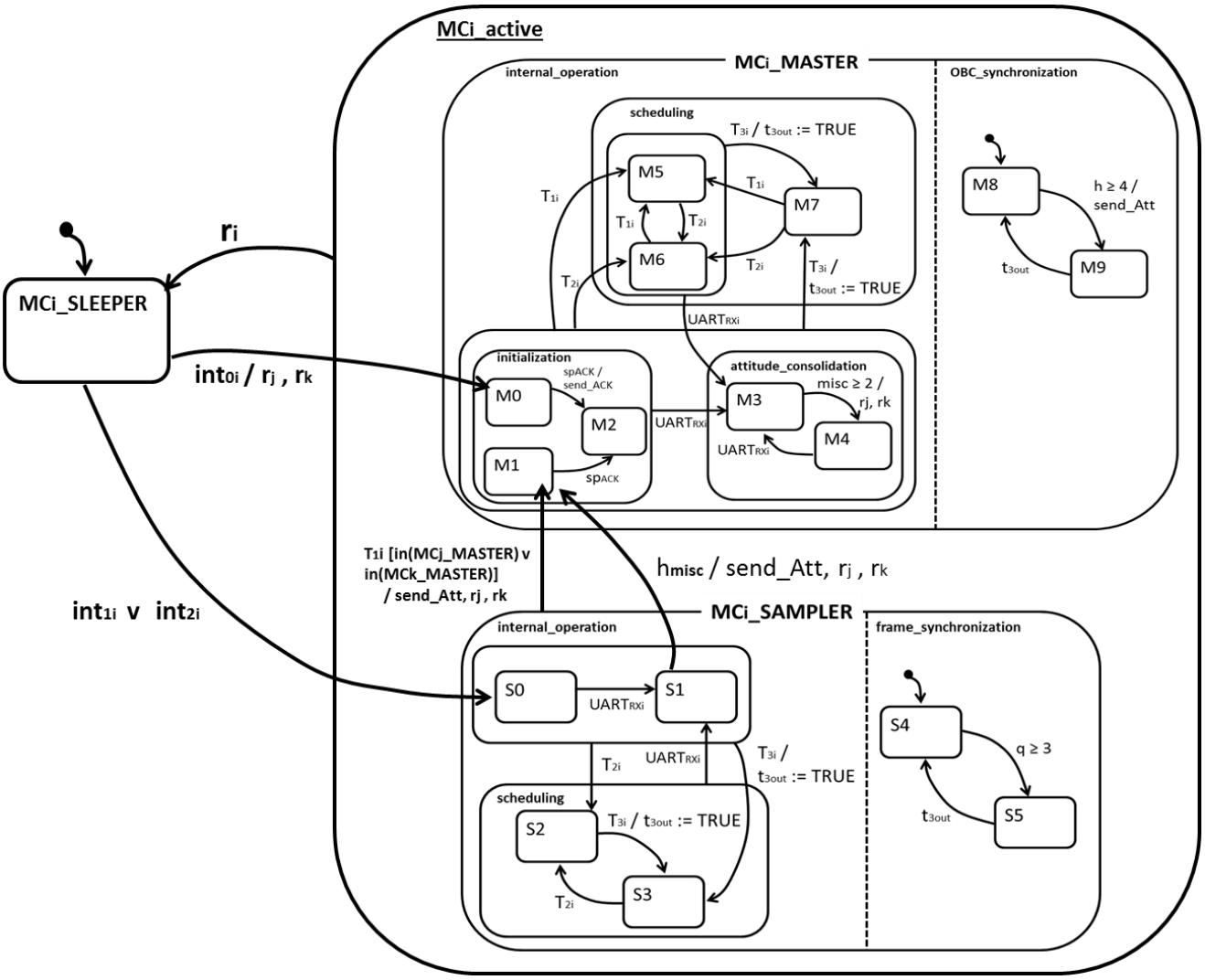


Figura 29 – Modelo em Statechart para o comportamento de cada microcontrolador.

Até esse ponto foi descrita, por meio de um formalismo, a lógica de estados da operação do SDATF. No entanto, essa descrição representa apenas os disparos de transições e os modos de operação dos microcontroladores ao longo do tempo. Resta descrever as ações realizadas por cada microcontrolador em cada estado.

Pode-se, a partir do modelo proposto em statechart, construir um modelo de simulação para cada estado, como se cada estado fosse um cenário da operação do microcontrolador.

O próximo capítulo é dedicado à descrição da construção desse modelo, que instancia o grafo aqui apresentado, fazendo com que esse grafo determine qual estado será simulado em um dado momento.

Capítulo 5

Aspectos de Construção do Modelo de Simulação do SDATF

A construção de um modelo em statechart para o SDATF foi o passo inicial na obtenção de um modelo de simulação que possa ser usado para validação. Além de ser uma representação não ambígua do sistema, o statechart apresentado permite a construção de um modelo de simulação dividido em estados.

O statechart criado facilita a organização do modelo de simulação, tornando-o mais simples. Sem ele, o modelo a ser criado seria muito mais complexo, visto que suas operações deveriam ser genéricas o suficiente para cobrir todas as combinações de estados (cada microcontrolador pode assumir três papéis em um dado momento, sendo que os papéis de MASTER e SAMPLER possuem várias tarefas).

A ideia, portanto, é construir um modelo de simulação no ambiente Simulink[1] dividido por subsistemas, onde cada subsistema permita a simulação de um estado do statechart. O statechart apresentado é instanciado usando o pacote *Simulink StateFlow*. As condições geradas na simulação alimentam o modelo statechart, gerando transições para novos estados, e o estado ativo em um dado momento determina qual subsistema irá definir os sinais de saída do modelo. Os aspectos relativos à construção desse modelo serão abordados nas próximas seções.

5.1 Simulação Híbrida

Em algumas aplicações encontradas na literatura é comum a construção de modelos de *Simulação Híbrida* (SAHBANI; PASCAL, 2000; MISHRA; PAPPAS; SOKOLSKY, ; ALUR et al., 2008). Esse tipo de modelo, representado na Figura 30, é composto por duas partes - uma parte formada por blocos genéricos, de acordo com a aplicação a ser modelada, que representam o comportamento dinâmico do sistema, e outra parte

[1] 1994-2015 The MathWorks, Inc.

representada por um diagrama de estados e transições, que controla quais blocos do modelo dinâmico estarão ativos em um dado momento.

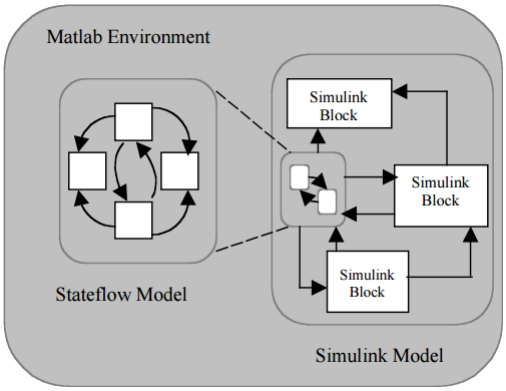


Figura 30 – Esquemático mostrando simulação híbrida no ambiente Simulink. No modelo representado, um diagrama de estados e transições construído no ambiente Simulink StateFlow interage com blocos comuns do ambiente Simulink.

Fonte: (SAHBANI; PASCAL, 2000)

O conceito de simulação híbrida foi utilizado na construção do modelo de simulação do SDATF. A parte do modelo que representa os estados e transições foi descrita no Capítulo 4, usando o formalismo Statechart. O ambiente Simulink possui um pacote voltado para a construção desse tipo de modelo - o *Simulink StateFlow*[2], que será abordado na Seção 5.2. Existem diversas ferramentas que podem ser utilizadas para simulação de um sistema como SDATF. A escolha do Simulink se deve à praticidade oferecida pelo pacote *Simulink StateFlow* na construção de um modelo com a sintaxe do Statechart.

A parte do modelo referente à dinâmica comportamental do SDATF foi construída com blocos comuns ao ambiente Simulink, e será abordada na Seção 5.3. Afim de estabelecer clareza nas descrições a seguir, as expressões "blocos do Simulink" ou "modelo dinâmico" serão utilizadas para se referir aos blocos aritméticos comuns ao ambiente Simulink que são usados para modelar o comportamento dinâmico do SDATF, em oposição ao diagrama construído no ambiente StateFlow, que descreve a lógica de transição entre os estados de cada microcontrolador.

A Figura 31 mostra o modelo Simulink construído para o SDATF. Deve ser observado que esse modelo baseia-se na mesma ideia de simulação híbrida mostrada na Figura 30,

[2] O Simulink StateFlow é uma ferramenta para construção de diagramas de transição entre estados. Ele foi usado para construir o modelo descrito no Capítulo 4, o qual foi desenvolvido usando o formalismo statechart. O StateFlow, portanto, é a ferramenta usada para construção do modelo, ao passo que statechart é o formalismo que define a sintaxe usada na representação desse modelo. Estabelecidos esses conceitos, as expressões "modelo statechart", "modelo construído no StateFlow"ou "diagrama construído no ambiente StateFlow"serão usadas nesse texto com o mesmo intuito para se referir à parte do modelo de simulação que descreve a lógica de transição entre os estados.

onde o modelo construído no ambiente StateFlow (em azul, à direita na figura) interage com os demais blocos Simulink.

Ao lado esquerdo da figura estão três instâncias de um modelo comportamental construído para o microcontrolador, além de um subsistema representando as medições dos sensores. O modelo construído no ambiente StateFlow é abordado na Seção 5.2 e o modelo dinâmico comportamental é descrito na Seção 5.3.

O diagrama construído no ambiente StateFlow considera valores de algumas variáveis para definir quais estados estarão ativos em um dado momento e se determinadas transições irão ocorrer ou não (variáveis de entrada do statechart). Essas variáveis podem ser internas (como os *timers* que definem o escalonamento de tarefas, por exemplo) ou externas (como *flags* que indicam recebimento de informação pelo canal UART, ou o valor do *heartbeat*).

No caso das variáveis externas, há uma interface entre o modelo em statechart e o modelo de blocos Simulink, na qual as condições de contexto, provenientes da simulação, são recebidas pelo modelo feito no ambiente StateFlow. Além disso, o modelo em statechart também permite variáveis de saída, de forma que esse modelo forneça sinais que determinam o comportamento dos blocos Simulink. Em outras palavras, o modelo contendo a lógica de estados e transições (construído no Simulink StateFlow) fornece informações que controlam o comportamento do modelo dinâmico (construído com blocos aritméticos do Simulink). Em contrapartida, as operações do modelo dinâmico determinam valores que podem disparar transições no diagrama StateFlow.

Exemplos de sinais produzidos pelo modelo dinâmico são as interrupções int_0, int_1 e int_2 referentes a cada microcontrolador, variáveis que indicam recebimento de dados pelo canal UART ($UART_{RX}$) e o valor corrente do *heartbeat*. Em contrapartida, o modelo em statechart fornece variáveis que controlam o comportamento do modelo Simulink. Conforme será abordado na Seção 5.3, o modelo que representa as ações executadas por cada microcontrolador foi dividido em subsistemas, onde cada subsistema executa a simulação de um estado definido no statechart. Dessa forma, foi atribuído um valor numérico inteiro exclusivo a cada estado definido na Figura 29 do Capítulo 4, sendo esse inteiro utilizado como um identificador do estado[3]. Três variáveis do tipo inteiro armazenam o identificador referente ao estado ativo em cada um dos microcontroladores, MC_i, MC_j e MC_k, de forma que essas variáveis são fornecidas pelo diagrama StateFlow ao modelo dinâmico no ambiente Simulink. Essa ideia está ilustrada na Figura 32.

[3] Estado SLEEPER = 0, M0 = 1, M1 = 2, e assim por diante.

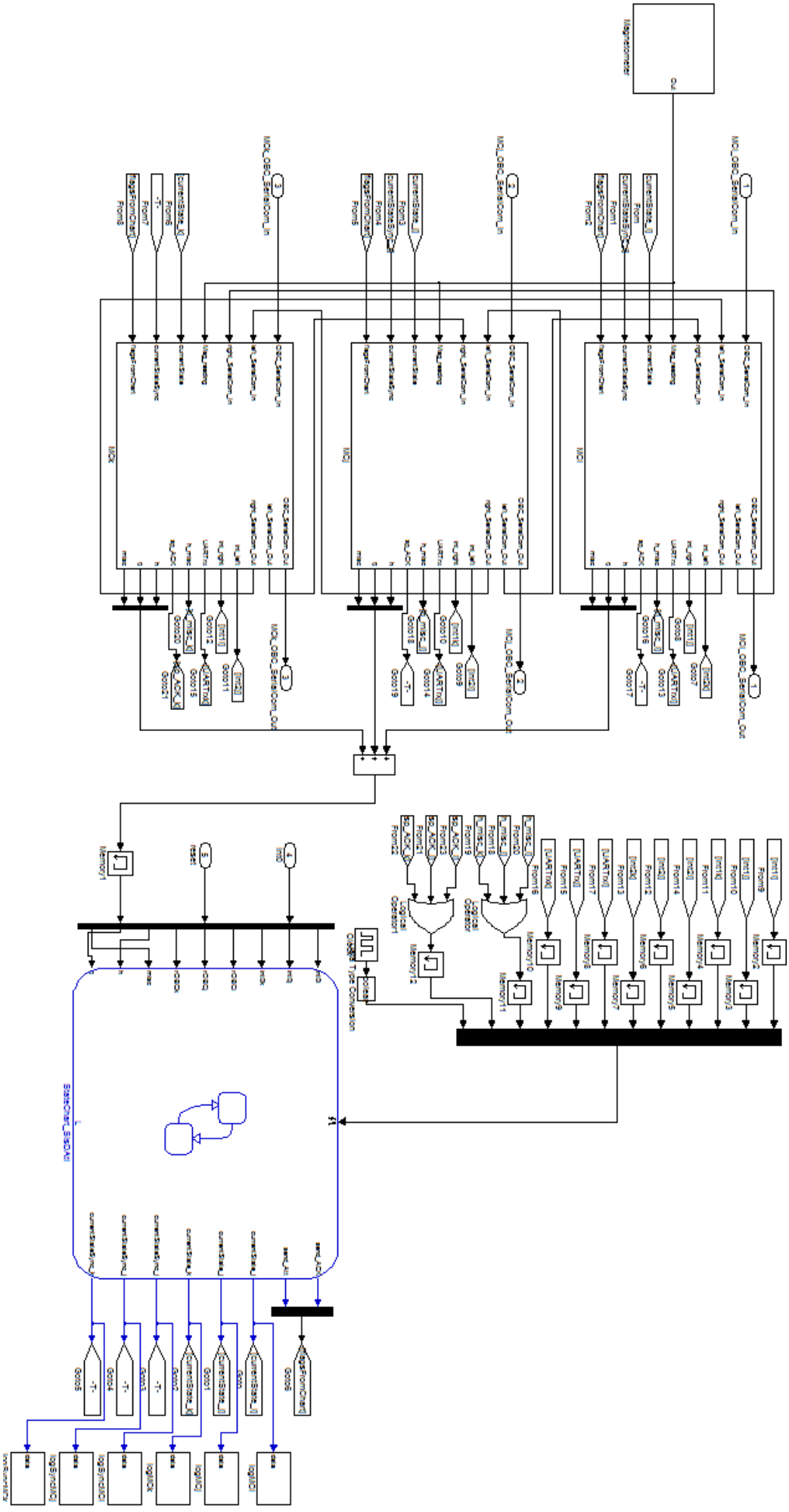


Figura 31 – Modelo híbrido de simulação do SDATF.

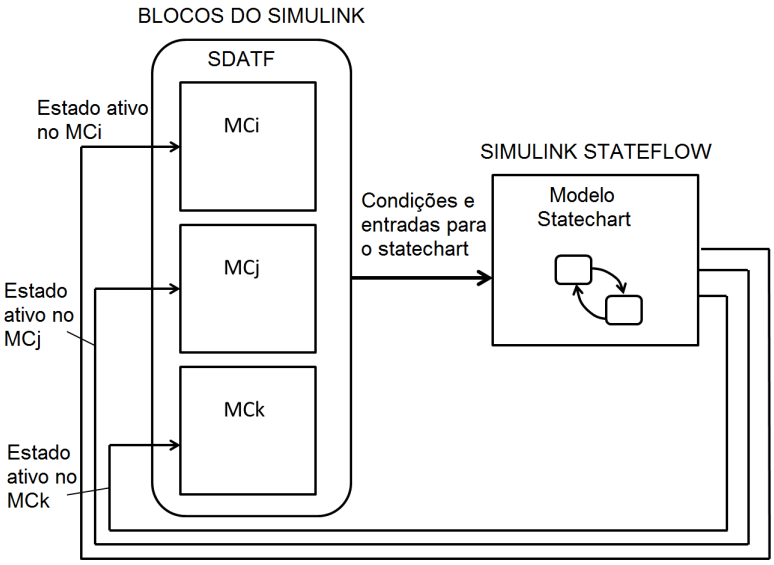


Figura 32 – Realimentação dos estados ativos de cada microcontrolador.

O modelo de simulação, portanto, utiliza os valores dos identificadores fornecidos pelo statechart para definir qual estado será simulado em um dado instante.

5.2 Simulink StateFlow

O *Simulink StateFlow* é um ambiente que permite a simulação baseada em grafo de estados e transições de um sistema lógico. Esse ambiente possui uma sintaxe baseada na definição dos Statecharts feita por D. Harel (HAREL, 1987), sendo expandida ao longo de novas versões, permitindo novas funcionalidades como tabelas de transições e tabelas-verdade.

O ambiente Simulink StateFlow foi utilizado para construir o diagrama de estados e transições do SDATF, tal como descrito no Capítulo 4. A Figura 33 mostra o diagrama construído.

Deve ser notado que a sintaxe definida nesse ambiente equivale à sintaxe dos diagramas statechart descritos no Capítulo 4 - estados em pontilhado indicam composição AND, ao passo que estados em linha contínua indicam composição exclusiva (XOR).

No nível hierárquico mais alto, o modelo construído possui dois elementos em composição AND. O primeiro componente representa o modelo de transições proposto para o SDATF, composto por três instâncias do modelo apresentado na Figura 29, sendo cada uma referente a um microcontrolador (MC_i, MC_j e MC_k). A Figura 34 mostra a porção da Figura 33 referente a um dos microcontroladores. Essa figura representa a lógica proposta na Figura 29, com as devidas adaptações necessárias para construção no ambiente *Simulink StateFlow*.

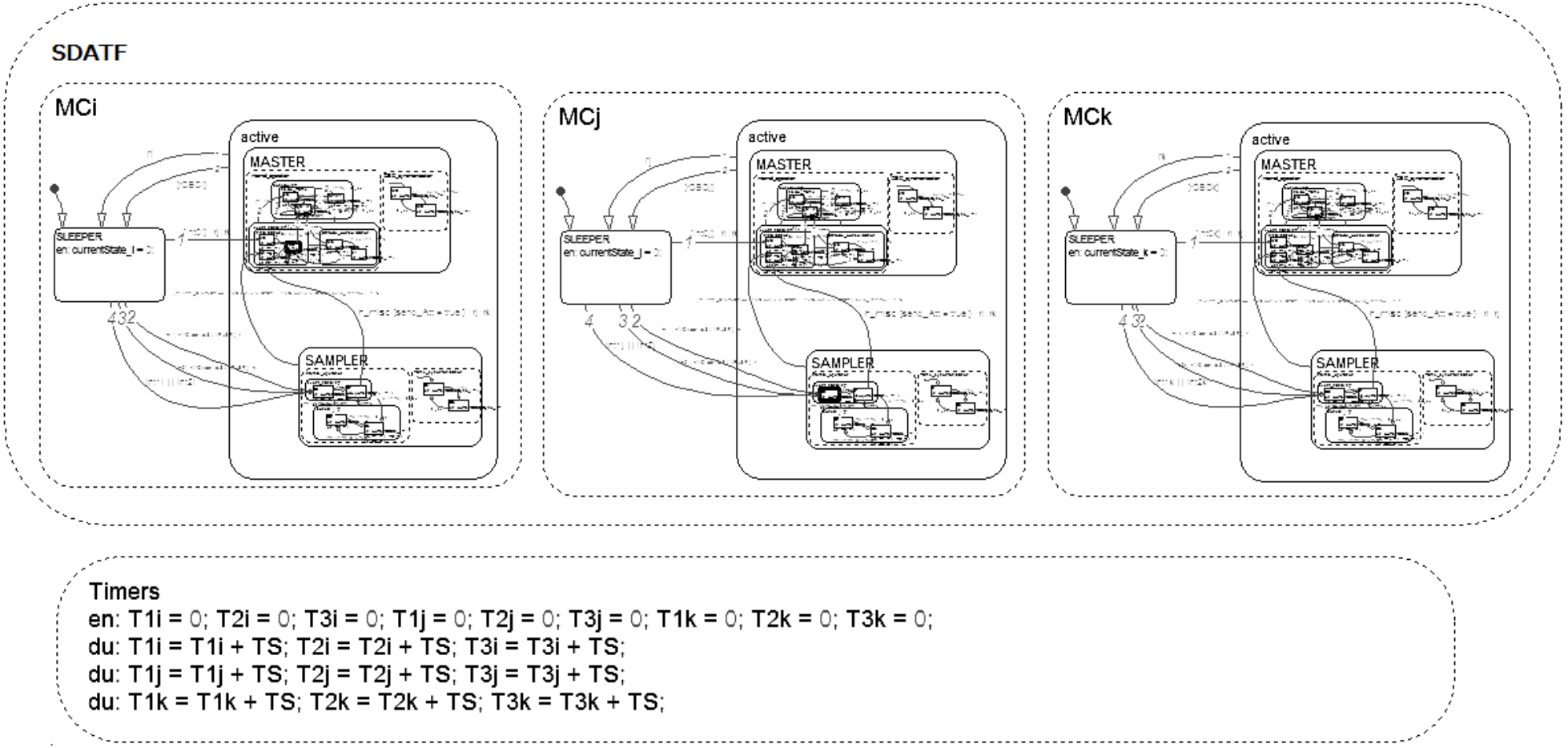


Figura 33 – Modelo de transições e estados do SDATF no StateFlow.

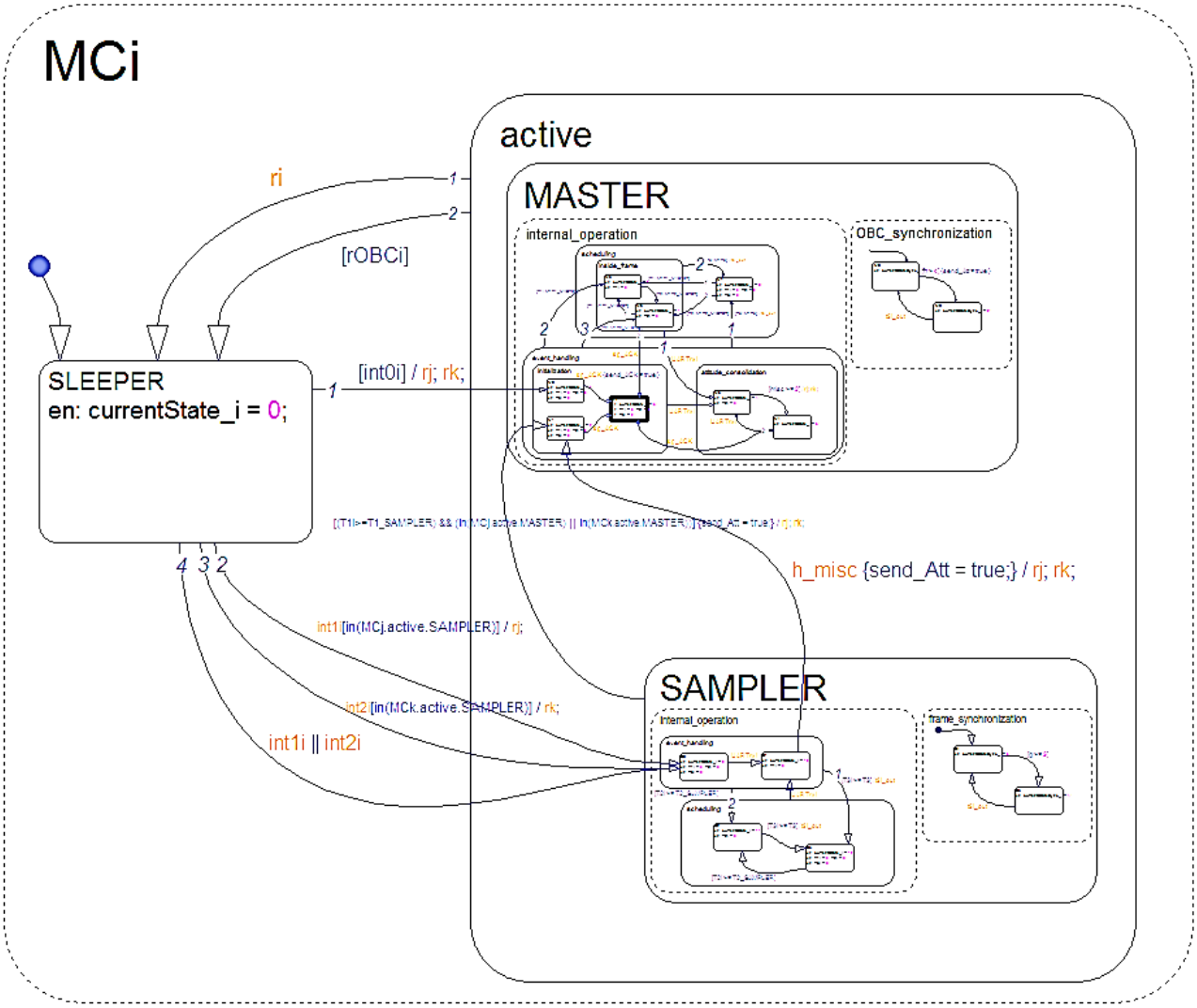


Figura 34 – Detalhamento do modelo construído no ambiente Stateflow referente a um microcontrolador do SDATF.

As figuras 35 e 36 apresentam os detalhes das figuras 33 e 34 referentes aos comportamentos do microcontrolador MC_i quando nos papéis de MASTER e SAMPLER, respectivamente. Essas figuras apresentam a construção no ambiente *Simulink StateFlow* das lógicas representadas anteriormente pelas figuras 27 e 28.

O outro elemento da Figura 33 foi criado para lidar com os *timers* envolvidos na aplicação (timers T_1, T_2 e T_3, para cada microcontrolador, i, j e k). O StateFlow permite incluir ações associadas a cada estado. Nessa aplicação foram utilizados dois tipos de ações:

- *en*, ou *entry*: São ações executadas uma única vez, apenas no ciclo de simulação no qual o estado foi ativado;
- *du*, ou *during*: São ações que são executadas a cada ciclo de simulação enquanto o estado estiver ativo.

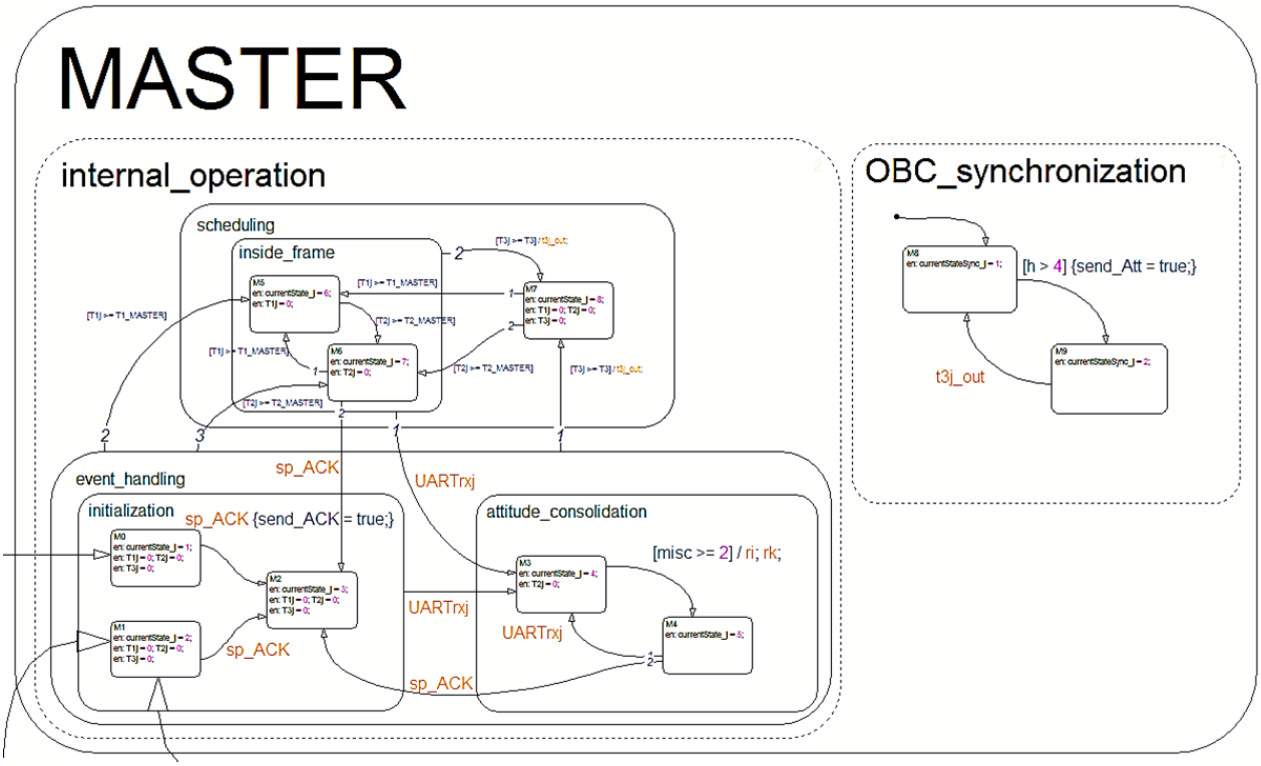


Figura 35 – Construção do modelo para o modo MASTER no ambiente *Simulink StateFlow*.

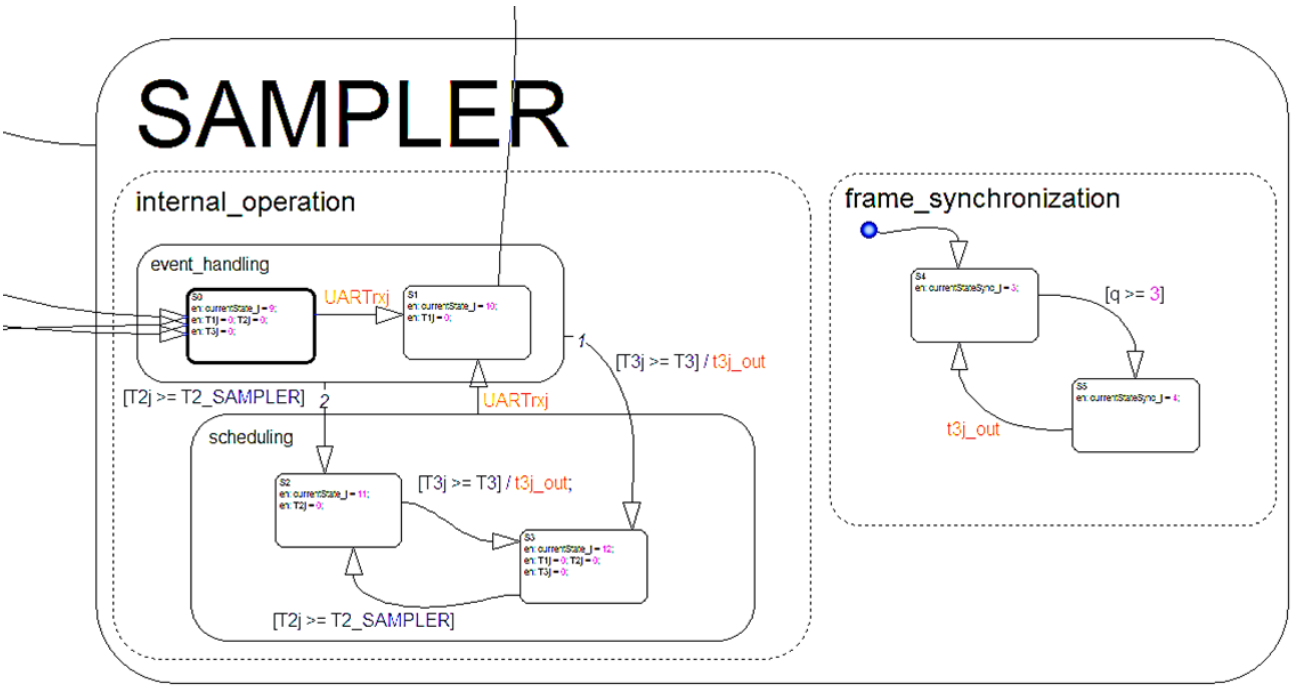


Figura 36 – Construção do modelo para o modo SAMPLER no ambiente *Simulink StateFlow*.

Como o elemento que lida com a inicialização e atualização dos timers está definido como um estado no maior nível de hierarquia em composição AND com o elemento do SDATF, esse estado permanece ativo durante toda a simulação. A inicialização dos timers foi definida como ação do tipo "en", sendo executada apenas uma vez (os timers podem ser zerados ao longo da simulação por meio de ações definidas em outros estados de menor hierarquia). A atualização desses timers é feita a cada ciclo da simulação (ações do tipo "du"), somando-se o período de amostragem (TS) a todos os timers.

5.3 Modelo Comportamental do Sistema

Na seção anterior foi abordada a construção do modelo no ambiente Simulink State-Flow, considerando o modelo proposto no capítulo anterior e a atualização dos timers. Nessa seção será abordada a parte do modelo referente às ações que ocorrem em cada estado.

A parte do modelo de simulação referente ao comportamento dinâmico do SDATF é composta por três instâncias do modelo comportamental para um microcontrolador e um sistema representando as medições de sensores.

Para o modelo dos sensores foram consideradas saídas constantes, de forma que o valor da atitude calculada possa ser conhecido. Dessa forma é possível identificar mais facilmente se houve algum erro nos resultados, uma vez que o objetivo da simulação é estudar a solução tolerante a falhas, e não o algoritmo para cálculo da atitude.

Os três microcontroladores são instâncias de um único modelo, visto que esses microcontroladores são programados com o mesmo software. Esse modelo leva em conta os subsistemas referentes a cada estado e os canais de comunicação serial. Como há comunicação serial de um microcontrolador com o MC à esquerda, à direita e com o computador de bordo do satélite, foram criados modelos dedicados à comunicação serial, sendo um transmissor e um receptor. Esses componentes foram instanciados na criação do modelo do microcontrolador.

A Subseção 5.3.1 descreve como o modelo do microcontrolador trata o chaveamento de estados, e a Subseção 5.3.2 trata do modelo de comunicação serial.

5.3.1 Modelo comportamental do microcontrolador

Conforme mencionado, o comportamento do SDATF foi simulado por meio de três instâncias de um modelo para o microcontrolador. Esse modelo recebe sinais provenientes do statechart, que contém a lógica de transições entre os estados de operação de cada microcontrolador, e fornece sinais com condições geradas em sua simulação.

A Figura 37 mostra o modelo para o microcontrolador. Sua interface possui três instâncias de receptores e três de transmissores (os modelos para transmissor e receptor

serão descritos na Subseção 5.3.2), referentes à comunicação serial com o computador de bordo do satélite (OBC), e com os microcontroladores à esquerda e à direita. Além da comunicação serial, o microcontrolador recebe alguns sinais provenientes do modelo em statechart e também fornece sinais a esse modelo, conforme já mencionado.

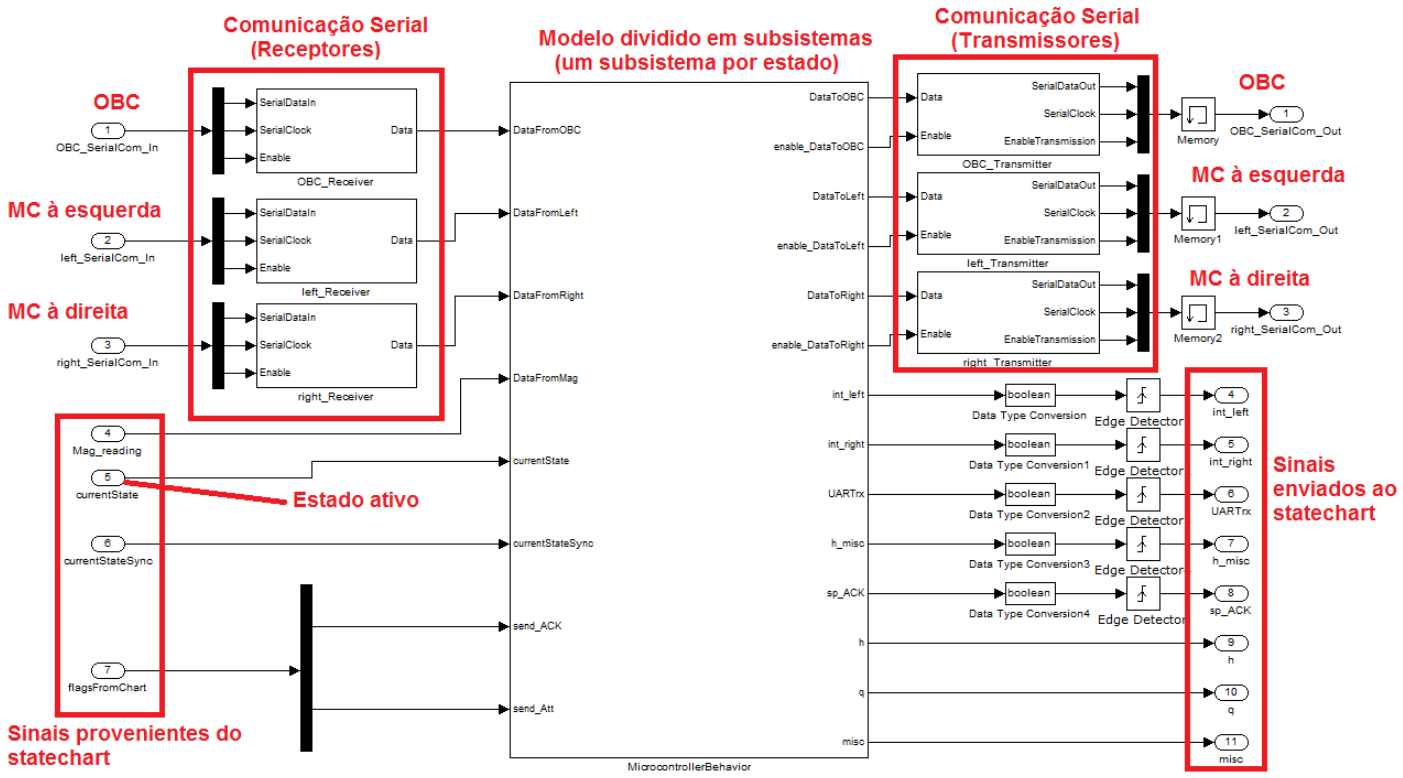


Figura 37 – Modelo de simulação do microcontrolador do SDATF.

Dentre os sinais recebidos do statechart está o número inteiro que indica o estado ativo. A parte central do modelo de simulação do microcontrolador, mostrado na Figura 37, é dividida em subsistemas, sendo que cada subsistema contém o modelo para simulação de um único estado. O microcontrolador recebe o identificador que indica o estado ativo e em função desse identificador determina qual subsistema estará habilitado em um dado momento, como mostrado na Figura 38. Nessa figura, a variável que armazena o estado ativo a cada instante é comparada com o identificador de cada estado possível. O resultado de cada comparação é uma variável booleana que é usada para habilitar a parte do modelo referente à simulação do respectivo estado.

Para cada estado, portanto, existe uma variável booleana que habilita o subsistema que contém sua operação, conforme mostrado na Figura 39. Essa figura mostra parte dos subsistemas e seus respectivos sinais de habilitação. Trata-se de uma lógica semelhante à estrutura *switch-case* utilizada em programação. Os modelos mostrados nas figuras Figura 38 e Figura 39 são internos ao subsistema indicado ao centro da Figura 37.

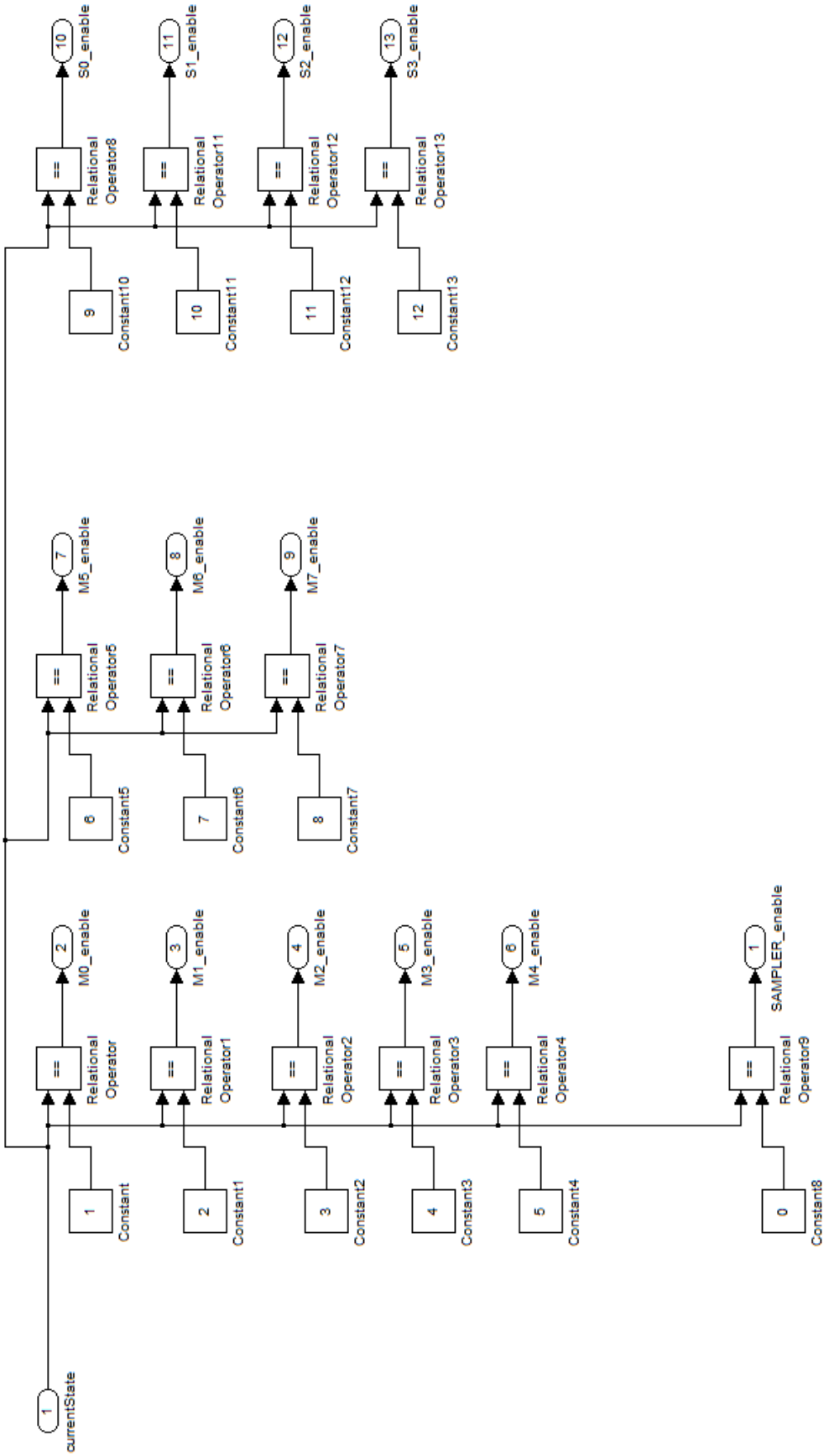


Figura 38 – Variáveis de habilitação determinadas em função do estado ativo.

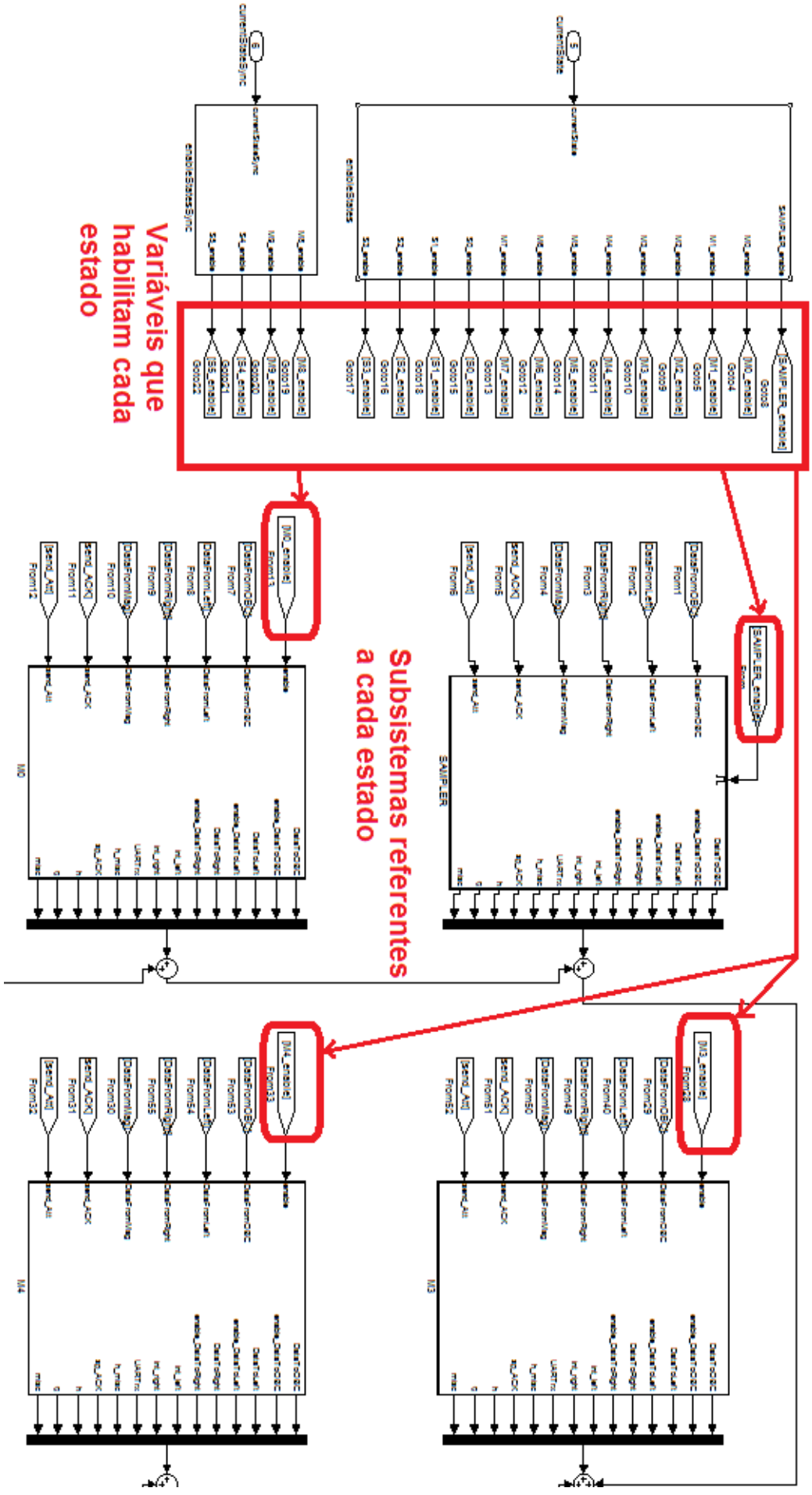


Figura 39 – Variáveis de habilitação para cada subsistema.

Em um dado instante, apenas um estado referente à operação interna do microcontrolador está habilitado. Dessa forma, a abordagem utilizada simplifica a construção do modelo, uma vez que utiliza o conceito de *dividir para conquistar*, permitindo que se construa um modelo para cada estado, e selecione o modelo ao longo do tempo em função do estado ativo.

As saídas de cada subsistema são consolidadas, resultando nas saídas do microcontrolador. Esses valores são somados, de forma que quando um subsistema não está habilitado, suas saídas são zeradas. Havendo apenas um subsistema habilitado em um dado momento, as saídas do microcontrolador serão equivalentes às desse subsistema.

5.3.2 Comunicação Serial

O modelo para comunicação serial leva em conta variáveis com representação em ponto flutuante de acordo com o padrão IEEE 754. O modelo genérico de comunicação serial converte o valor a ser transmitido em binário e transmite os dados um bit a cada passo da simulação.

No ambiente MatLab[4] dados em representação de ponto flutuante tem precisão dupla (64 bits) por padrão. Como a aplicação do sistema tolerante a falhas trabalha com precisão simples (32 bits), a informação é convertida do formato *double* para *single*, e em seguida expressa na forma binária. Após a transmissão, a informação é novamente convertida para ponto flutuante.

A modelagem é dividida em dois módulos - o transmissor e o receptor. A Figura 40 a seguir esquematiza a funcionalidade a ser modelada.

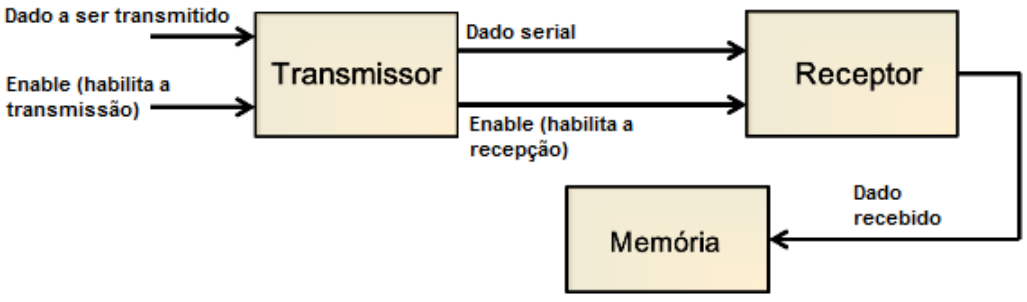


Figura 40 – Esquema mostrando interface dos modelos para comunicação serial.

Na transmissão serial, os sinais são convertidos do formato em ponto flutuante (decimal) para binário, ao passo que na recepção o processo inverso ocorre. Após converter a informação para um formato binário, ele deve ser transmitido e recebido bit a bit. A informação recebida, por fim, é novamente convertida para o formato decimal e memorizada.

[4] 1994-2015 The MathWorks, Inc.

A Figura 41 mostra o modelo em Simulink para o módulo transmissor. O modelo possui como entradas o valor a ser enviado e um sinal que habilita a transmissão, e como saídas o dado serial, um sinal de clock[5] e o sinal que habilita a recepção por parte do outro módulo. Quando a transmissão é habilitada, uma função implementada em Matlab transforma o valor da entrada em um vetor de bits e a cada ciclo de simulação um bit é transmitido, juntamente com um sinal de clock.

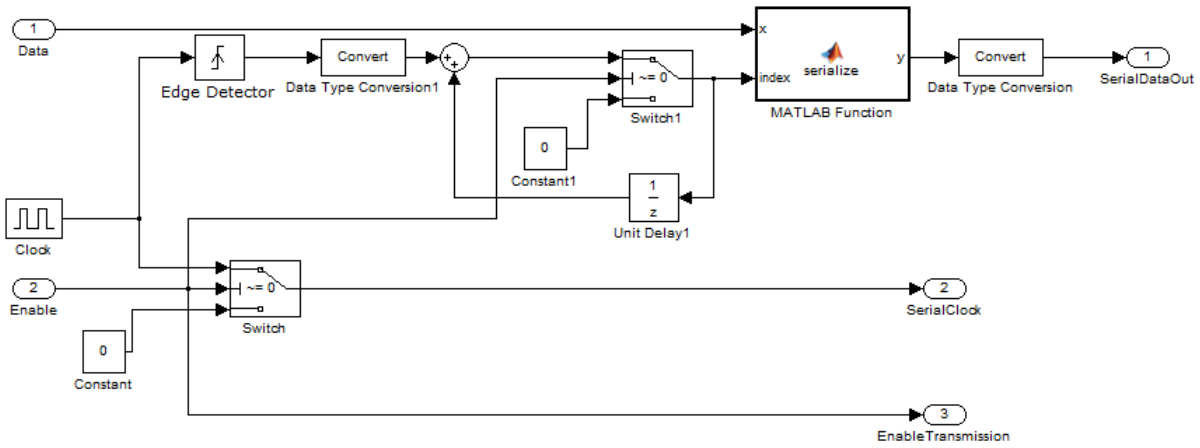


Figura 41 – Modelo Simulink para o módulo transmissor serial.

O modelo para o receptor é mostrado na A Figura 42. Esse modelo possui como entrada os sinais de saída do transmissor - o dado serial, o sinal de clock e uma entrada que habilita a recepção. Como saída o módulo fornece o dado recebido no formato de ponto flutuante.

Quando a recepção é habilitada, uma função lê a informação na entrada a cada borda de subida do clock, acumulando esses valores até montar um vetor de 32 bits. Quando isso ocorre, o vetor é transformado no formato decimal e o subsistema *HoldValue* mantém o valor obtido até que uma nova recepção ocorra.

Os gráficos da Figura 43 mostram os sinais observados em uma simulação da comunicação serial.

Nessa simulação, com duração de 200 ciclos, dois valores foram transmitidos, um positivo $(31,45)$ e outro negativo $(-20,06)$. No primeiro ciclo o valor positivo foi disponibilizado na entrada e a transmissão foi habilitada por 64 ciclos, sendo desabilitada em

[5] O módulo de comunicação serial utilizado no SDATF é de natureza assíncrona. O sinal de *clock* utilizado nesse modelo, portanto, não modela um sinal na interface física. Esse sinal de clock está relacionado ao passo discreto da simulação, e é utilizado apenas para emular o clock interno do microcontrolador na comunicação via UART. Como o bloco Simulink que fornece esse sinal é baseado no passo de simulação, o modelo que simula uma comunicação síncrona apresenta o mesmo comportamento de um modelo para a simulação assíncrona. Em outras palavras, devido à própria natureza da simulação de passo fixo, pode-se dizer que os modelos operam em sincronismo.

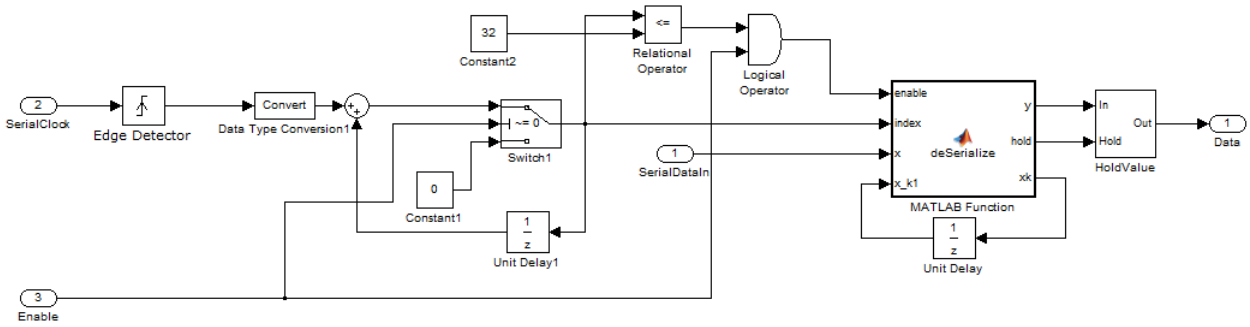


Figura 42 – Modelo Simulink para o módulo receptor serial.

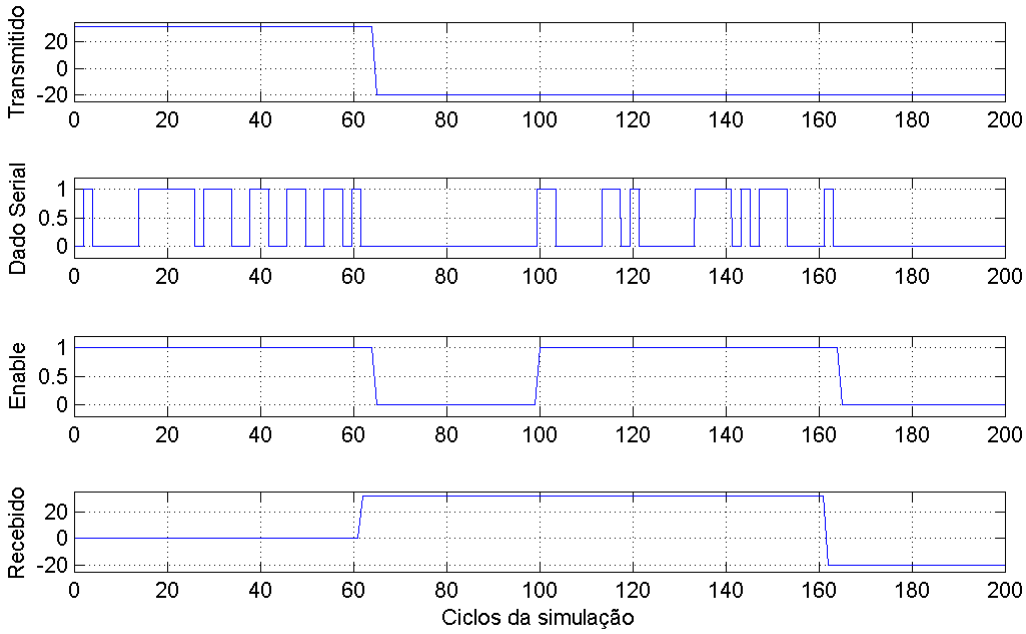


Figura 43 – Simulação da transmissão serial de dados.

seguida. O valor foi recebido e mantido na saída. No ciclo 100 o a transmissão do valor negativo foi iniciada. Assim que esse valor foi recebido, a saída foi atualizada e mantida.

Deve ser observado que o valor recebido, disponível no sinal de saída do módulo receptor, é equivalente ao sinal transmitido, entrada do transmissor, acrescido de um atraso, inerente ao processo de transmissão serial.

5.4 Simulação de Cenários

O modelo apresentado na Figura 31 foi utilizado para simular o comportamento do SDATF. A simulação foi configurada com passo discreto e período de $0,0001$ segundo, o

suficiente para capturar corretamente o comportamento dinâmico do sistema [6]. As variáveis referentes ao estado ativo dos três microcontroladores foram registradas, permitindo identificar o papel e a tarefa executada por cada microcontrolador em qualquer instante.

A seguir serão apresentados três cenários nos quais o modelo foi utilizado para simulação. Nesses três cenários foram simulados quatro segundos de operação do SDATF [7]. O objetivo ao simular esses cenários é demonstrar o escalonamento descrito no Capítulo 3, bem como o chaveamento de papéis entre os microcontroladores.

Conforme mencionado na descrição do sistema, a operação se inicia quando o computador de bordo do satélite dispara a interrupção int_0 em um dos microcontroladores. Nesse caso, foi simulada uma inicialização no instante $t = 0,5s$, por meio do disparo de int_{0i}. Dessa forma, espera-se que o microcontrolador MC_i seja inicializado no papel de MASTER, inicializando MC_j para assumir o papel de SAMPLER.

Dito isso, os três cenários podem ser descritos da seguinte forma:

- Cenário 1 - Operação normal. O objetivo dessa simulação é mostrar como o SDATF deve se comportar na ausência de falhas;
- Cenário 2 - Falha no SAMPLER. O objetivo é demonstrar o chaveamento entre microcontroladores quando o SAMPLER envia informação corrompida ao MASTER;
- Cenário 3 - Falha no MASTER. O objetivo é demonstrar o chaveamento entre microcontroladores quando o MASTER sofre variação no sinal de *heartbeat*.

A Figura 44 mostra o Cenário 1, um cenário de operação normal do SDATF, na qual nenhuma falha ocorre. Com o uso das variáveis registradas foram identificados os eventos principais que ocorreram ao longo da simulação.

Na figura é possível observar que o microcontrolador MC_i foi de fato inicializado no papel de MASTER próximo ao instante $t = 0,5s$. Em seguida, o microcontrolador MC_j é inicializado como SAMPLER e a rotina de tarefas entre esses dois agentes é iniciada. O MASTER envia o sinal de *heartbeat* ao SAMPLER respeitando o tempo limite de $220ms$, enquanto o SAMPLER envia a atitude ao MASTER periodicamente, dentro do limite de $240ms$. Após enviar o valor de *heartbeat* pela quarta vez, o MASTER finalmente envia a última atitude válida ao computador de bordo do satélite e uma nova janela de sincronismo é iniciada.

Outro cenário simulado (Cenário 2) foi o de envio da atitude inválida por parte do SAMPLER. Como o MC_j é inicializado como SAMPLER na operação normal, foi forçado

[6] Deve ser lembrado que a dinâmica de operação do SDATF é baseada em um escalonamento cujos *timers* têm durações de 220 ms, 240 ms, 250 ms, 280 ms e 1 segundo. Portanto, o passo discreto de simulação com período de $0,0001$ segundo é o suficiente para que o comportamento do sistema seja capturado durante a simulação.

[7] A duração de quatro segundos na simulação se deve apenas ao fato de que, para demonstrar a operação do SDATF, é necessário simular um tempo de operação de alguns segundos, visto que a janela de sincronismo tem duração de um segundo.

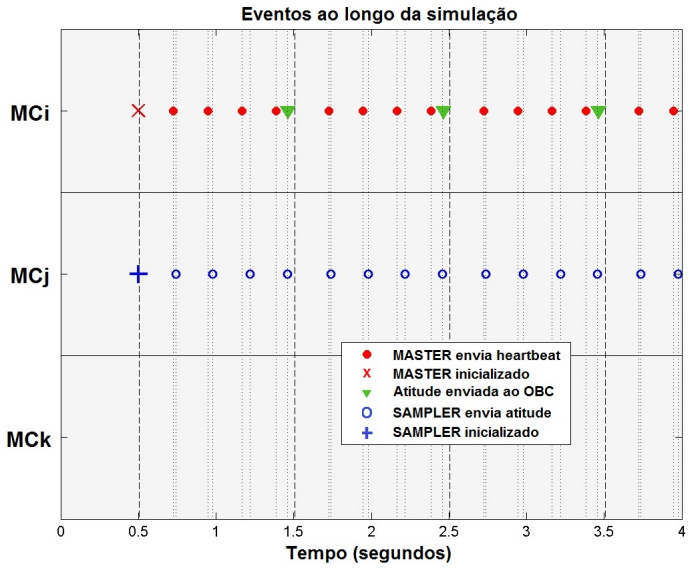


Figura 44 – Simulação do SDATF em operação normal (Cenário 1).

um valor inválido na atitude calculada por esse microcontrolador. A Figura 45 mostra os eventos identificados na simulação desse cenário.

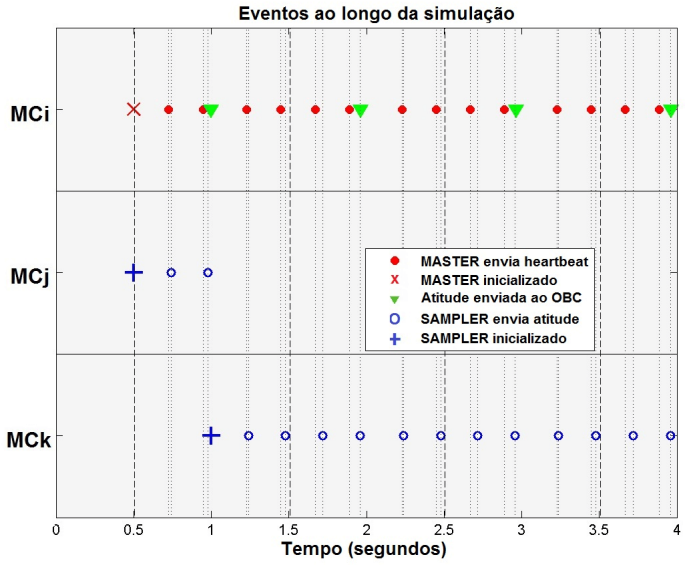


Figura 45 – Simulação do SDATF quando o SAMPLER envia atitude inválida (Cenário 2).

Nesse cenário o MC_j envia a atitude ao MC_i, que está no papel de MASTER, duas vezes. Após o envio da atitude inválida pela segunda vez, o MC_i envia um sinal de reset ao MC_j, enviando em seguida a última atitude válida (calculada pelo MC_i ao ser inicializado) ao OBC e inicializando o MC_k para assumir o papel de SAMPLER. Isso ocorre em um instante próximo a $t = 1s$. A partir desse momento, o SDATF segue

operando o MC_i como MASTER e o MC_k como SAMPLER.

O terceiro cenário simulado (Cenário 3) foi o de falha no MASTER. Nesse caso o valor de referência do *heartbeat* do MC_i foi alterado. O resultado da dessa simulação é mostrado na Figura 46.

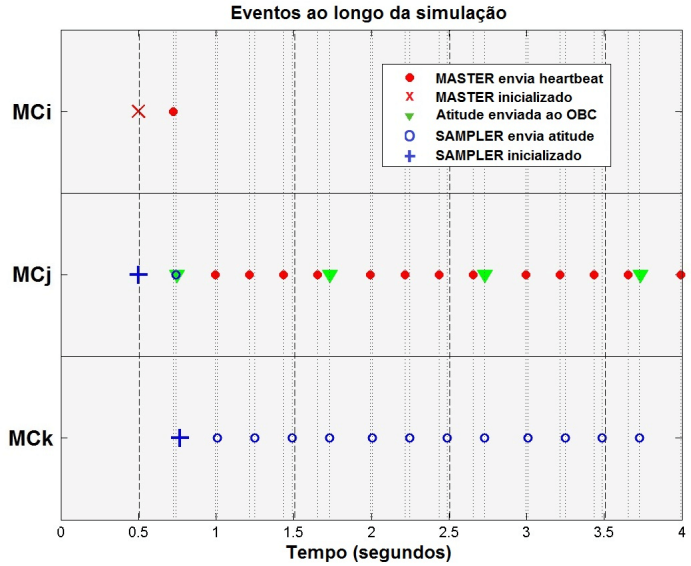


Figura 46 – Simulação do SDATF quando o MASTER envia um valor incorreto de *heartbeat* (Cenário 3).

Assim como nos outros cenários, o MC_i é inicializado como MASTER no instante $t = 0,5s$. Durante a operação, no entanto, o MC_i envia um valor incorreto de *heartbeat* ao MC_j, este até então no papel de SAMPLER. Ao identificar a falha, o MC_j envia um sinal de reset ao MC_i, enviando em seguida a última atitude válida ao OBC. Quando isso ocorre, o MC_j passa a assumir o papel de MASTER e inicializa o MC_k para ser o novo SAMPLER. A operação do SDATF segue normalmente a partir desse momento.

Nos três cenários descritos, o sistema foi inicializado em $t = 0,5s$. Assim sendo, espera-se que a atitude deve ser fornecida dentro dos limites de tempo definidos pela janela de sincronismo de 1 segundo, ou seja, até $t = 1,5s$, $t = 2,5s$, $t = 3,5s$, e assim sucessivamente. Deve ser observado que nos três cenários simulados o tempo limite de 1 segundo foi respeitado para envio da atitude consolidada.

Nos cenários 2 e 3 foi necessário forçar falhas em regiões específicas do modelo para executar a simulação. No entanto, as falhas do tipo SEU são de natureza estocástica no tempo, sendo que a região de memória afetada também é de caráter aleatória.

No próximo capítulo serão descritos os experimentos realizados com o intuito de se validar o SDATF. Para isso foi criado um procedimento para injeção de falhas em instantes de tempo e regiões do modelo aleatórios.

Capítulo 6

Validação por Experimentos com Injeção de Falhas, Resultados e Análises

Após a apresentação e contextualização do problema de falhas do tipo *Single Event Upset*, a descrição do SDATF e a construção de um modelo de simulação do comportamento desse sistema, resta a validação da solução tolerante a falhas.

No capítulo anterior, na Seção 5.4, foram apresentados cenários de simulação do SDATF. Nos cenários onde ocorrem falhas, essas falhas tiveram de ser forçadas em regiões ou dados específicos do modelo. No entanto, as falhas do tipo SEU são de natureza estocástica. Nas seções a seguir será apresentado o procedimento para a validação do SDATF por meio de injeção de falhas aleatórias. Os aspectos relativos a esse procedimento serão descritos e o processo genérico para validação por modelagem será resumido. Por fim, os resultados e análises das simulações serão apresentados.

6.1 Experiência prévia na validação do SDATF

Conforme já mencionado, esforços anteriores foram feitos com o intuito de se validar o SDATF. Torres (TORRES, 2013), em seu trabalho de mestrado, realizou experimentos envolvendo injeção de falhas em um protótipo, por meio de um método denominado *Coding Emulating Upset* (CEU). Esse método consiste na injeção de falhas por meio de rotinas de interrupção criadas para alterar dados com disparos aleatórios, emulando a ocorrência SEU (VELAZCO; REZGUI; ECOFFET, 2000). Em seu trabalho, o autor utilizou um protótipo baseado nos microcontroladores PIC24FJ64GA002, da *Microchip*. Ele mapeou os registradores sensíveis à ocorrência de *Single Event Upsets* e realizou experimentos com injeção de falhas, analisando o comportamento do SDATF.

Os experimentos realizados por Torres, no entanto, indicaram baixa disponibilidade do sistema. A disponibilidade é calculada em função de duas métricas - *Mean Time To Failure* (MTTF) e *Mean Time To Repair* (MTTR). O MTTF equivale ao tempo médio, a partir da inicialização do sistema ou de uma recuperação, até que ocorra uma falha

que deixe o sistema inoperante. Já o MTTR é o tempo médio gasto pelo sistema para se recuperar após sofrer uma falha. A disponibilidade D do sistema, portanto, é calculada por

$$D = \frac{MTTF}{MTTF + MTTR}. \tag{7}$$

Rosa (ROSA, 2014), em sua monografia de graduação, se propôs a investigar as causas da baixa disponibilidade indicada por Torres. Ele diagnosticou que Torres havia utilizado uma taxa de falhas por segundo bastante superior ao esperado na baixa órbita e realizou os experimentos considerando taxas mais realistas, que indicaram boa disponibilidade do sistema.

O intuito desse capítulo é apresentar os experimentos de injeção de falhas usando a nova abordagem, baseada em modelagem, considerando esses trabalhos anteriores como referência.

6.2 Mapeamento das Regiões Sensíveis

Um passo importante em uma campanha de injeção de falhas é o mapeamento das regiões sensíveis a falhas do tipo SEU, que consiste em definir as regiões de interesse onde essas falhas serão injetadas.

Na literatura define-se falha como um desvio no comportamento correto do sistema ao longo de seu ciclo de vida, ao passo que o erro é a manifestação dessa falha, ou seja, é a presença detectável ou perceptível da falha (AVIŽIENIS et al., 2004). Velazco (VELAZCO; REZGUI; ECOFFET, 2000) definiu três tipos de erro que são de interesse em um procedimento de injeção de falhas do tipo SEU:

- *Tolerated Errors*: são erros gerados pela inversão de bits em regiões de memória que não são relevantes à aplicação no momento da ocorrência. Os dados corrompidos podem ser sobrescritos em algum momento ou mesmo nunca serem utilizados;

- *Result Errors*: ocorrem quando a inversão de bits causa resultados incorretos em uma ou mais saídas do sistema;

- *Sequence Loss*: é o efeito mais crítico da inversão de bits. Ocorre quando essa inversão resulta em um travamento ou incapacidade do processador de gerar as saídas. Nesse caso o processador precisa ser reiniciado.

No trabalho de Torres, foram mapeados os registradores sensíveis à ocorrência de SEU, observando erros do tipo *Result Error* e *Sequence Loss*. A abordagem por modelagem e

simulação, no entanto, permite trabalhar em um nível mais alto de abstração, ou seja, permite mapear tarefas sensíveis à ocorrência de erros, uma vez que apenas o comportamento do sistema é modelado.

Considerando os estados definidos na Seção 4.5, foram definidas as tarefas que poderiam ser afetadas por uma inversão de bits. Uma vez levantada essa lista, as falhas podem serão injetadas na simulação, podendo alterar a saída dos blocos Simulink responsáveis por essas tarefas. A seguir essas tarefas são listadas para cada estado.

Valores salvos em memória

Os modelos de simulação para os microcontroladores utilizam variáveis, salvas no contexto do MatLab (*workspace*), que na aplicação real seriam salvas em memória volátil. Entre essas variáveis, estão os valores limite dos timers, o valor corrente do *heartbeat* (h), a última atitude válida, o número de vezes que a atitude foi enviada do SAMPLER para o MASTER (q) e o contador que indica o número de valores inválidos enviados pelo SAMPLER ($misc$). Todas essas variáveis são sensíveis à inversão de bits por *Single Event Upset*, e por isso estão sujeitas à injeção de falhas durante a simulação.

SLEEPER

O microcontrolador no estado SLEEPER apenas aguarda por uma interrupção que o faça assumir outro papel, não havendo ações nesse estado, portanto a ocorrência de um SEU em um microcontrolador no estado SLEEPER provavelmente ocasionará um erro do tipo *Tolerated Error*. Mesmo que isso não ocorra, o microcontrolador sempre recebe um sinal de reset antes de ser "acordado" para assumir outro papel. Sendo assim, não há injeção de falhas no estado SLEEPER.

SAMPLER

O papel de SAMPLER é dividido em seis estados, que vão de S_0 a S_5 (ver Seção 4.5). As regiões sensíveis do modelo para os estados S_0 a S_3 estão listadas na Tabela 1. Os estados S_4 e S_5 são referentes à sincronização do frame de 1 segundo, não havendo tarefa executada (apenas inicialização de timers). Nesses estados o microcontrolador está sujeito apenas à injeção de falhas nas variáveis do contexto (valores salvos em memória).

Os elementos listados na Tabela 1 constituem todos os blocos, relacionados às funções de cada estado no modo SAMPLER, que influenciam as saídas do microcontrolador.

MASTER

O papel de MASTER é dividido em dez estados, que vão de M_0 a M_9 (ver Seção 4.5). As regiões sensíveis do modelo para os estados M_0 a M_7 estão listadas na Tabela 2.

Tabela 1 – Regiões sensíveis do modelo para os estados no papel de SAMPLER.

Estado	**Tarefa**	**Região sensível do modelo**
S_0	Cálculo da atitude	Valores obtidos dos sensores; Valores de atitude calculados.
	Envio de sinal de confirmação ao MASTER, indicando que está ativo	Sinal enviado ao MASTER.
S_1	Verificar valor recebido pela UART	Valor na saída do canal UART.
	Verificar valor do *heartbeat*	Valor de h.
S_2	Envio da atitude calculada ao MASTER	Valores de atitude a serem enviados.
	Incremento do número de vezes que a atitude foi enviada ao MASTER (q)	Valor a ser salvo em q.
S_3	Inicialização dos timers e de q	Valores dos timers e de q.

Tabela 2 – Regiões sensíveis do modelo para os estados no papel de MASTER.

Estado	**Tarefa**	**Região sensível do modelo**
M_0	Cálculo da atitude	Valores obtidos dos sensores; Valores de atitude calculados.
	Inicialização do *heatbeat*	Valor de h.
	Inicialização do SAMPLER	Valor do tempo máximo de espera por resposta do SAMPLER.
M_1	Envio da última atitude válida ao OBC	Valores de atitude a serem enviados.
	Inicialização do *heatbeat*	Valor de h.
	Inicialização do SAMPLER	Valor do tempo máximo de espera por resposta do SAMPLER.
M_2	Envio do sinal de *Acknowledgement* ao OBC	Caractere de *Acknowledgement*.
M_3	Verificar dados recebidos do SAMPLER	Valores recebidos pela UART.
M_4	Cálculo da atitude	Valores de atitude a serem enviados.
	Inicialização do *heatbeat*	Valor de h.
	Inicialização do SAMPLER	Valor do tempo máximo de espera por resposta do SAMPLER.
M_5	Envio do sinal de *heartbeat* ao SAMPLER	Valor de h.
M_6	Envio da última atitude válida ao OBC	Valores de atitude a serem enviados.
	Inicialização do *heatbeat*	Valor de h.
	Inicialização do SAMPLER	Valor do tempo máximo de espera por resposta do SAMPLER.
M_7	Inicialização do *heartbeat*	Valor de h.

Os estados M_8 e M_9 são referentes à sincronização do frame de 1 segundo e envio da última atitude válida ao OBC. Nesses estados o microcontrolador está sujeito à injeção de falhas nas variáveis de contexto (valores salvos em memória) e às falhas injetadas nos módulos de comunicação serial.

Os elementos listados na Tabela 2 constituem todos os blocos, relacionados às funções de cada estado no modo MASTER, que influenciam as saídas do microcontrolador.

Módulos de comunicação serial

Os módulos de comunicação serial também podem sofrer efeitos por inversão de bits. No caso de um canal UART, por exemplo, há registradores envolvidos que armazenam o valor a ser transmitido (ou recebido) em *buffer*, como também há registradores que controlam a transmissão ou recepção dessa informação. No modelo de transmissor foram injetadas falhas no dado a ser transmitido, e no sinal que indica qual posição do vetor de bits será enviada no ciclo de simulação corrente. No modelo do receptor foram injetadas falhas no valor recebido e armazenado, bem como no índice que indica a posição do bit recebido.

6.3 Injeção de Falhas

Uma vez definidas as regiões do modelo sensíveis a falhas do tipo SEU, criou-se uma lista de blocos e variáveis relativos a essas regiões. Considerou-se que qualquer região do modelo possui a mesma probabilidade de sofrer uma inversão de bits, visto que a distribuição de falhas do tipo SEU no espaço é homogênea. Essa afirmação é razoável para fins experimentais, uma vez que o circuito é muito pequeno se comparado a distâncias no espaço nas quais se observa diferenças na densidade de partículas de alta energia. A cada evento de falha, portanto, um bloco ou variável da lista era sorteado.

Embora a distribuição da ocorrência de SEU no espaço físico do circuito seja homogênea, o mesmo não se pode dizer da distribuição dessas falhas no tempo. A seguir será descrito o modelo de injeção de falhas utilizado.

6.3.1 Modelo estocástico para ocorrência de SEU

Seja um intervalo de tempo t, dividido em subintervalos de duração Δt, cuja duração é curta o suficiente para se considerar nula a probabilidade de ocorrência de mais de um *Single Event Upset*. A probabilidade de ocorrência de um SEU no intervalo Δt pode ser definida por

$$P(n = 1; \Delta t) = \lambda \Delta t \tag{8}$$

onde λ é a taxa média de ocorrências por unidade de tempo, dada em SEU/s, e n é o número de SEU ocorridos no intervalo Δt. Considerando que $P(n \geq 2; \Delta t) = 0$ (dado que o intervalo Δt é curto o suficiente), a probabilidade de não ocorrer nenhum SEU ao longo de Δt é dada por

$$P(n = 0; \Delta t) = 1 - P(n = 1; \Delta t) = 1 - \lambda \Delta t. \quad (9)$$

Suponha que a probabilidade de não ocorrer nenhum SEU no intervalo de tempo t, dada por $P(n = 0; t)$, seja conhecida, e deseja-se saber a probabilidade da não ocorrência de SEU no intervalo $t + \Delta t$, ou seja, $P(n = 0; t + \Delta t)$.

Um fato importante é que, devido à natureza dos fenômenos *Single Event Upset*, pode-se dizer que tais fenômenos são eventos independentes. Em outras palavras, a ocorrência ou não de um SEU no intervalo de t segundos não influencia a probabilidade de uma nova ocorrência nos próximos Δt segundos. Assim sendo, a probabilidade $P(n = 0; t + \Delta t)$ pode ser definida pelo produto

$$P(n = 0; t + \Delta t) = P(n = 0; t)P(n = 0; \Delta t) = P(n = 0; t)(1 - \lambda \Delta t). \quad (10)$$

Reorganizando a Equação 10:

$$\frac{P(n = 0; t + \Delta t) - P(n = 0; t)}{\Delta t} = -\lambda P(n = 0; t). \quad (11)$$

Se o intervalo Δt for levado ao limite em que tende a zero, tem-se que

$$\frac{dP(n = 0; t)}{dt} = -\lambda P(n = 0; t). \quad (12)$$

A solução para a equação diferencial em 12 é dada por

$$P(n = 0; t) = Ce^{-\lambda t} \quad (13)$$

onde C é uma constante. Para $t = 0$, a probabilidade de não ocorrer nenhum SEU deve ser de 100%, ou seja, $P(n = 0; 0) = 1$. Dessa forma, tem-se que $C = 1$, e portanto

$$P(n = 0; t) = e^{-\lambda t}. \quad (14)$$

A Equação 14 indica que a probabilidade da não ocorrência de falhas do tipo *Single Event Upset* decai exponencialmente ao longo do tempo, e decai tão rápido quanto maior for a taxa de falhas λ. Essa distribuição de probabilidade equivale à distribuição de Poisson. De fato, essa distribuição caracteriza um modelo estocástico que está de acordo com a observação de falhas do tipo SEU na baixa órbita terrestre (GOKA et al., 1991).

6.3.2 Taxa de injeção de falhas

Para a execução de uma campanha de injeção de falhas, deve-se considerar um valor de referência para a taxa de falhas λ. Considerando o procedimento para validação por modelagem e simulação, caso não haja alguma medição experimental anterior, esse valor pode ser definido em função de alguma especificação ou requisito da solução proposta.

Como o procedimento aqui proposto dá continuidade ao trabalho de Torres e Rosa (TORRES, 2013; ROSA, 2014), a taxa de falha definida nos experimentos anteriores será utilizada como referência, o que permite comparar a abordagem por modelo à injeção de falhas em protótipo.

Faure e Velazco (FAURE; VELAZCO; PERONNARD, 2005) publicaram um trabalho contendo resultados de experimentos envolvendo a exposição de um processador à radiação e a comparação do número de falhas observadas em relação ao valor obtido em uma campanha de injeção de falhas. Por meio da comparação, definiu-se a taxa λ que resultava em um comportamento mais próximo ao observado na presença de radiação. A taxa encontrada em relação à área exposta é de 2,3 $SEU/s.cm^2$.

Com base nesse valor, Torres realizou experimentos considerando $\lambda = 2,3$ SEU/s no protótipo baseado nos microcontroladores PIC24FJ64GA002, o que resultou em baixa disponibilidade do SDATF. Após investigação, Rosa observou que a área física sensível é significativamente inferior à considerada por Torres.

O PIC24FJ64GA002 possui 0,7 cm de largura e 3,5 cm de comprimento, o que define uma área de 2,45 cm^2. Essa área, no entanto, é a área do encapsulamento do microcontrolador. O circuito integrado ocupa aproximadamente 1/5 dessa área, conforme mostrado na Figura 47. Dividindo a área de 2,45 cm^2 por 5, temos a área do circuito integrado equivalente a 0,49 cm^2.

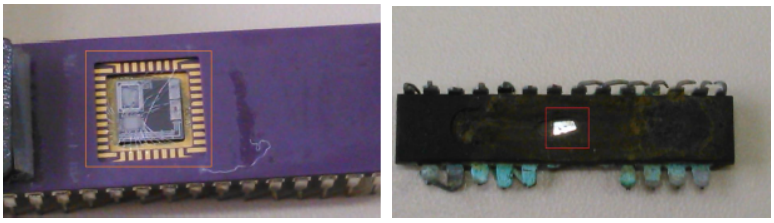


Figura 47 – Área do circuito integrado em relação ao encapsulamento.

A memória disponível no PIC24FJ64GA002 equivale a 72192 bytes (memória de programa e de dados). Após uma análise, Rosa fez um levantamento dos registradores sensíveis à inversão de bits e considerou 160 registradores como alvos de falhas. Cada registrador dentre os 160 possui 2 bytes, totalizando 320 bytes. Isso representa cerca de 0,44 % da memória disponível no microcontrolador. Dessa forma, da área de 0,49 cm^2 ocupada pelo circuito integrado, a área sensível à ocorrência de erros (relevante para a aplicação) é de 0,002156 cm^2 em cada microcontrolador.

Como o SDATF é composto por três microcontroladores, multiplica-se esse valor por três, e obtém-se uma área sensível de 0,006468 cm^2. A taxa de falhas λ a ser utilizada nos experimentos é obtida multiplicando-se a área sensível pela taxa de referência calculada por Faure e Velazco:

$$\lambda = 2,3 \ \frac{SEU/s}{cm^2} \times 0,006468 cm^2 \approx 0,0148 \ SEU/s. \tag{15}$$

Deve ser observado que o valor obtido para λ foi definido em função de três fatores - a taxa

de referência obtida experimentalmente por Faure e Velazco, o microcontrolador utilizado e o número de registradores sensíveis à inversão de bits. Esse último fator foi definido por Rosa em seu trabalho, e depende da aplicação programada em um protótipo. Para o caso de futuras mudanças na aplicação, caso se deseje executar nova validação, pode-se considerar os valores da aplicação anterior acrescidos de uma margem de segurança.

6.3.3 Geração do vetor de falhas

Na Seção 6.2 foram definidas regiões do modelo sensíveis a inversão de bits. Seja uma lista de m blocos ou variáveis, $b_1, b_2, ..., b_m$, possíveis alvos na injeção de falhas. A partir dessa lista de blocos sensíveis, da distribuição de probabilidade e da taxa de falhas λ, define-se um vetor de falhas a serem injetadas na simulação do SDATF.

O vetor de falhas é definido por uma lista de tuplas (B_k, T_k), onde B_k é o bloco (ou variável) do modelo que sofrerá a k-ésima falha da simulação no instante de tempo T_k.

A Figura 48 mostra o gráfico de $P(n = 0; t)$, descrita pela Equação 14, em função do tempo t, para a taxa de falhas de $\lambda = 0,0148$ SEU/s.

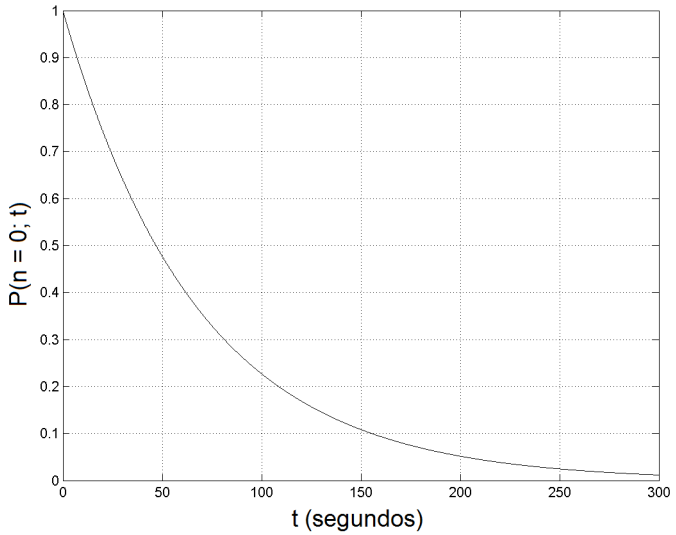


Figura 48 – Probabilidade da não ocorrência de uma falha ao longo do tempo com $\lambda = 0,0148$ SEU/s.

Esse gráfico pode ser entendido como a probabilidade da não ocorrência de uma falha a partir de um instante inicial $t = 0$. Para se obter a tupla que caracteriza a primeira falha, dada por (B_1, T_1) deve-se sortear um dos blocos sensíveis e um tempo, considerando a distribuição de Poisson com o valor de λ especificado.

Para seleção de B_1, sorteia-se um número inteiro N, com distribuição homogênea[1] no intervalo que vai de 1 a m. O bloco B_1 será o bloco sorteado b_N da lista de blocos $b_1, b_2, ..., b_m$. Para se obter T_1, sorteia-se um número real R com distribuição homogênea entre 0 e 1. A partir do número sorteado, obtém-se o tempo equivalente a R no gráfico da Figura 48. Esse é o instante a partir do qual a falha ocorre (para um tempo inferior a esse instante, não há ocorrência de falhas). A Figura 49 mostra esse procedimento para um valor arbitrário $R = 0,6554$.

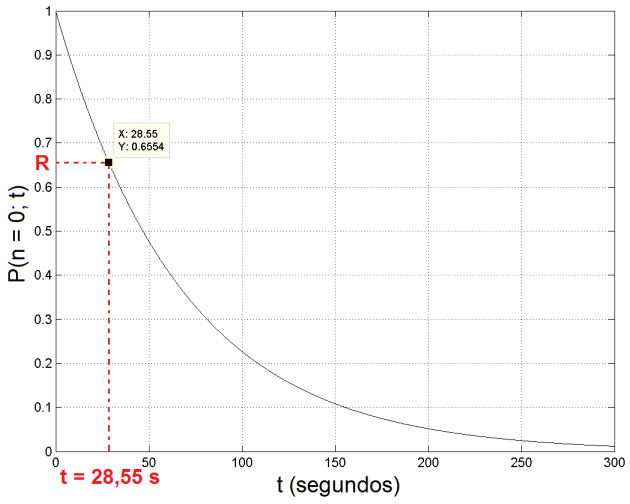


Figura 49 – Obtenção do tempo de ocorrência da falha a partir de um instante inicial.

Para $R = 0,6554$ e $\lambda = 0,0148$ SEU/s, o tempo encontrado é $T_1 \approx 28,55$ s. Esse procedimento consiste no mapeamento de uma distribuição homogênea para a distribuição de Poisson. Para isso, basta igualar o valor R sorteado à probabilidade de não ocorrer falha a partir de um instante inicial, dada por $P(n = 0; T_1)$. Considerando a Equação 14, tem-se que

$$R = e^{-\lambda T_1} \implies T_1 = -\frac{ln(R)}{\lambda}. \tag{16}$$

A partir da tupla obtida (B_1, T_1), deve-se obter a próxima falha (B_2, T_2), e assim sucessivamente. Genericamente, uma vez obtida a falha (B_k, T_k) obtém-se a próxima falha (B_{k+1}, T_{k+1}). Lembrando que a ocorrência de falhas do tipo SEU podem ser vistas como eventos probabilísticos independentes, pode-se repetir o procedimento utilizado para se

[1] Para esse experimento considerou-se que cada bloco ou variável da lista de alvos possui a mesma probabilidade de sofrer uma falha. Essa consideração é adequada ao experimento, visto que a área sensível do circuito é muito pequena em relação ao espaço no qual a densidade de partículas se altera de forma significativa. Dessa forma, a probabilidade de ocorrência de SEU no espaço pode ser modelada por uma distribuição homogênea.

obter (B_k, T_k) na obtenção de (B_{k+1}, T_{k+1}), obtendo-se T_{k+1} a partir do tempo sorteado somado ao tempo da falha anterior, T_k. Em outras palavras, a definição do tempo de ocorrência da falha $k+1$ equivale a obter o tempo de ocorrência de uma falha a partir do instante $t = 0$, deslocando o resultado do tempo para a ocorrência da k-ésima falha. Esse procedimento se repete até o tempo limite da simulação.

6.3.4 Falhas que afetam mais de um bit

Eventualmente, partículas de alta energia podem afetar mais de um bit de uma vez. A probabilidade de ocorrência de falhas desse tipo é significativamente inferior à de inversão de um único bit.

A Figura 50 mostra a probabilidade de ocorrência de erros em um experimento (MAIZ et al., 2003) realizado em células SRAM com tecnologia de fabricação de 130 nm e 90 nm.

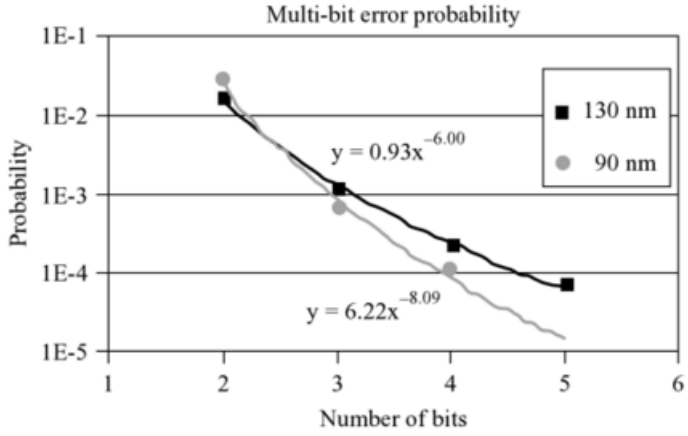


Figura 50 – Probabilidade de ocorrência de falhas em mais de um bit.

Fonte: Retirado de (MUKHERJEE, 2011, p. 32). Originalmente publicado em (MAIZ et al., 2003)

Observa-se que a probabilidade de um evento por partículas de alta energia afetar três ou mais bits em um registrador é muito inferior se comparada à possibilidade de apenas um bit ser afetado. No entanto, a variação de dois bits pode ocorrer com frequência superior a 1 % dos eventos ocorridos.

Frente a esses fatos, a ocorrência de falhas em mais de um bit deve ser considerada na injeção de falhas em um protótipo. No entanto, ao se trabalhar com modelagem a nível de comportamento, não há uma relação direta entre os registradores do dispositivo e os sinais sobre os quais as falhas são injetadas. Isso porque as falhas são injetadas por funcionalidade da aplicação. Em outras palavras, um sinal do modelo de simulação que sofre a falha eventualmente estará relacionado a diversos registradores do dispositivo físico.

Essa abstração configura mais uma vantagem na abordagem de validação por modelagem e simulação.

6.3.5 Modelagem da falha

O procedimento para injeção de uma falha é simples. A partir de um bloco Simulink, o qual gera o valor em que se deseja inserir uma falha, conecta-se outro bloco, denominado *Fault_Injector*, responsável por alterar o valor de saída do bloco alvo a partir de um instante de tempo específico. Isso é mostrado na Figura 51.

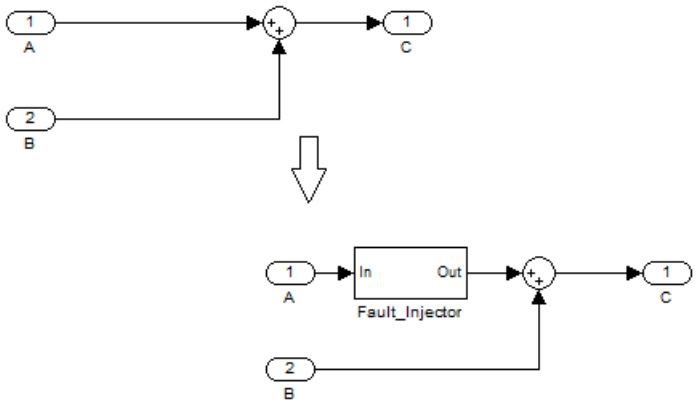


Figura 51 – Injeção de falha no bloco "A".

O bloco *Fault_Injector*, conforme mostrado na Figura 52, converte o sinal sobre o qual deve ser injetada a falha em um valor de ponto flutuante de 32 bits. Além disso ele utiliza o bloco *Clock*, como referência do tempo de simulação, para injeção da falha no instante correto. Por meio de uma função implementada em MatLab, sorteia-se um valor entre 0 e 31, referente a qual bit será invertido. O bit sorteado é invertido no instante definido para a falha, e o sinal é convertido novamente para o seu tipo original.

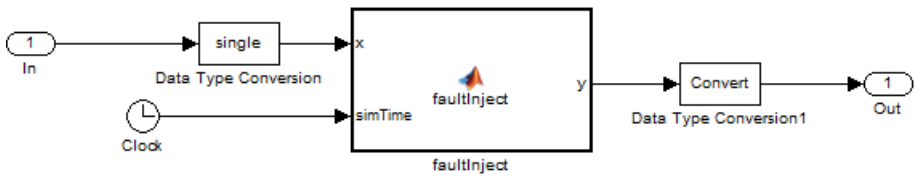


Figura 52 – Lógica interna ao bloco *Fault_Injector*.

Para injeção de falhas em variáveis definidas no *workspace* do MatLab, referentes aos valores armazenados em memória, utiliza-se uma função em MatLab que monitora o tempo corrente da simulação. Quando algum tempo do vetor de falhas é atingido, a função inverte um dos bits da variável alvo.

Esse processo, tanto para os blocos do modelo Simulink, quanto para as variáveis do *workspace*, é uma espécie de "instrumentação" do modelo de simulação. Essa intervenção agrega um custo computacional ao modelo, tornando a simulação mais lenta. No entanto, é importante ressaltar que essa intervenção não afeta o passo da simulação, visto que esse valor é fixo. Isso constitui uma vantagem sobre o método de injeção de falhas *Coding Emulating Upset*, pois na abordagem CEU a rotina de tratamento de interrupção é executada juntamente com a aplicação, podendo afetar o comportamento do sistema tolerante a falhas.

6.4 Resumo do procedimento utilizado

O fluxograma da Figura 53 mostra, de forma resumida, o procedimento para validação baseada em modelagem de uma solução tolerante a falhas.

O procedimento tem início com os requisitos que definem a solução tolerante a falhas. Esses requisitos são utilizados na elaboração de um modelo formal baseado em estados e transições (elaborado no Capítulo 4 usando statechart) e na construção de um modelo de simulação do comportamento do sistema (modelo Simulink abordado no Capítulo 5).

Utilizando um valor de referência para a taxa de falhas (Subseção 6.3.2) é realizada a campanha de simulação com injeção de falhas, composta pelos seguintes passos:

- ❏ Geração aleatória do vetor de falhas (Subseção 6.3.3);
- ❏ Criação de uma instância do modelo de simulação;
- ❏ Injeção de falhas no modelo (usando o modelo de falha descrito na Subseção 6.3.5);
- ❏ Execução da simulação.

Após a execução da simulação, os resultados são coletados e analisados. A partir do cálculo da disponibilidade, julga-se a validade da solução. Caso os resultados não sejam satisfatórios, deve-se investigar as causas da baixa disponibilidade e executar o procedimento novamente.

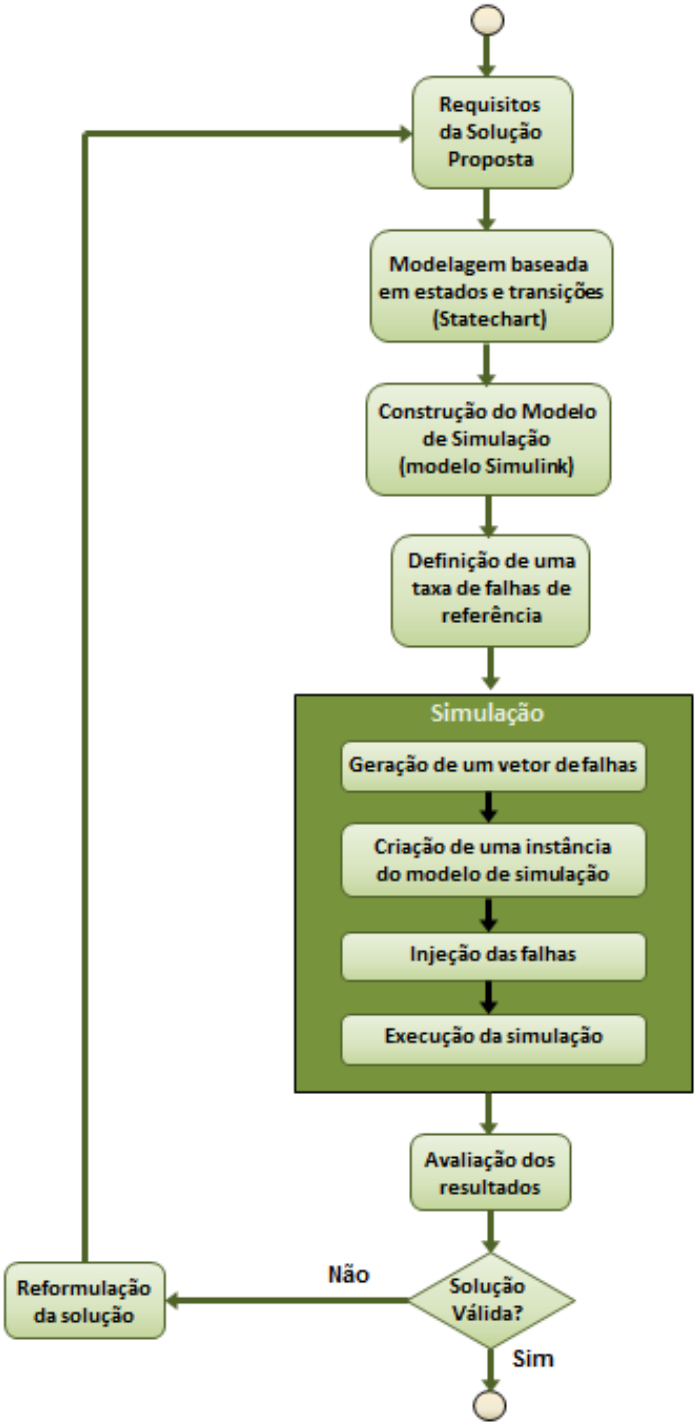


Figura 53 – Resumo do procedimento de validação.

6.5 Resultados e Análises

Os experimentos com injeção de falhas no modelo foram executados de uma forma semelhante ao procedimento realizado por Rosa no protótipo do SDATF, usando o método CEU (ROSA, 2014). O intuito de tal semelhança é permitir a comparação dos resultados.

Foram injetadas falhas no modelo construído para simular o comportamento do SDATF, conforme descrito nas Subseções 6.3.3 e 6.3.5. A cada envio da atitude ao computador de bordo do satélite (OBC), os valores foram registrados juntamente com o tempo de simulação em que o envio ocorreu. As simulações foram feitas sob os seguintes aspectos:

- O passo da simulação foi fixado em $0,0001$ s. Esse passo é o suficiente para capturar todo o comportamento dinâmico dos microcontroladores em operação. Conforme já mencionado, a dinâmica de operação do SDATF é baseada em um escalonamento cujos *timers* têm durações de 220 ms, 240 ms, 250 ms, 280 ms e 1 segundo;
- A simulação foi realizada com a taxa de falhas definida na Subseção 6.3.2, ou seja, $\lambda = 0,0148$ SEU/s. Também foram realizadas simulações com taxas de falhas múltiplas a esse valor, a saber: $0,37$ SEU/s (25 vezes), $0,74$ SEU/s (50 vezes) e $1,11$ SEU/s (75 vezes);
- Cada simulação foi feita com duração de uma hora de operação (3600 segundos), tal como no trabalho de Rosa, para que os resultados pudessem ser comparados;
- Para cada valor de λ a simulação foi executada 6 vezes, e a disponibilidade média foi considerada[2];
- Foram usados valores constantes para a atitude a ser enviada com o intuito de observar possíveis erros no valor final. Em cada simulação foi contabilizado o número de valores incorretos de atitude enviados ao OBC, considerando o erro mínimo detectável de 1 % do valor esperado.[3].

A Figura 54 mostra a distribuição dos papéis assumidos pelos microcontroladores MC_i, MC_j e MC_k ao longo dos 3600 segundos de simulação (em uma das seis execuções), para a taxa de falhas de $\lambda = 0,0148$ SEU/s.

Observa-se que, ao longo de uma hora de operação houveram poucos chaveamentos de papéis entre os microcontroladores. Apenas os microcontroladores MC_i e MC_j estiveram

[2] Uma das desvantagens da abordagem por modelagem e simulação é o alto custo computacional envolvido ao simular operações de longa duração e com altas taxas de falhas. Esse é um fator limitante ao número de execuções, ou seja, o número de amostras do processo para o tratamento estatístico. No entanto, conforme será observado nos resultados deste experimento, a execução para cada valor de λ 6 vezes foi o suficiente para se obter intervalos de confiança estreitos o suficiente para a validação do procedimento proposto neste trabalho.

[3] O erro mínimo detectável foi escolhido de forma arbitrária, apenas para estabelecer um limite prático entre os valores válidos ou corrompidos de atitude, visto que pequenas variações nos valores podem ocorrer devido a operações com ponto flutuante sem que haja falhas do tipo SEU.

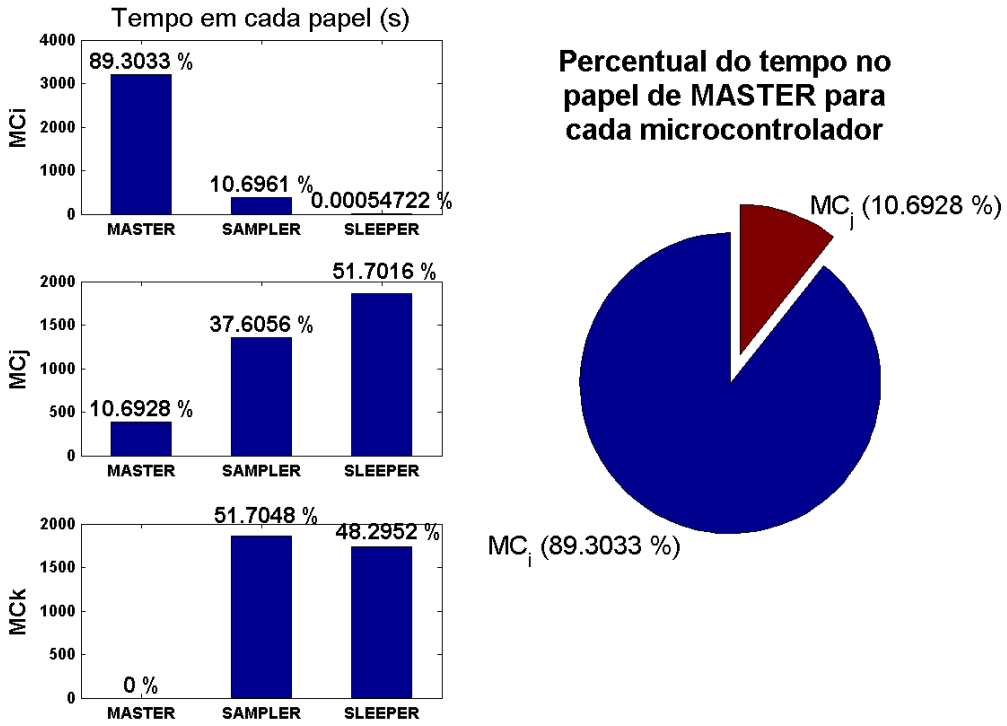


Figura 54 – Distribuição dos papeis de cada microcontrolador para $\lambda = 0,0148$ SEU/s.

no papel de MASTER, sendo que o MC_i esteve nesse papel na maior parte do tempo (quase 90 % do tempo simulado).

Para esse valor de λ a disponibilidade foi de 100 %, com 0 % de erro na atitude enviada ao OBC, em todas as simulações.

A Figura 55 mostra a distribuição de papéis ao longo dos 3600 segundos em uma das simulações realizadas com $\lambda = 0,37$ SEU/s, ou seja, uma taxa de falhas 25 vezes superior à taxa de referência.

Com o aumento de λ observa-se um aumento no chaveamento de papéis. Nessa simulação o microcontrolador MC_k, que antes não havia assumido o papel de MASTER, esteve nesse papel por cerca de 23 % do tempo de simulação, no entanto o MC_i ainda esteve nesse papel por mais da metade do tempo.

Para $\lambda = 0,37$ SEU/s a disponibilidade média do SDATF foi de aproximadamente 99,94 %, em um intervalo de confiança[4] entre 99,91 % e 99,98 % a um nível de confiança de 95 %. O percentual médio de erro nos valores enviados ao OBC foi de 1,51 %.

Para uma taxa de falhas 50 vezes maior ($\lambda = 0,74$ SEU/s), a disponibilidade média foi de aproximadamente 99,88 %, em um intervalo de confiança entre 99,85 % e 99,91 % a um nível de confiança de 95 %. Para esse valor λ o percentual médio de valores incorretos

[4] Detalhes a respeito do cálculo do intervalo de confiança são descritos no Apêndice A

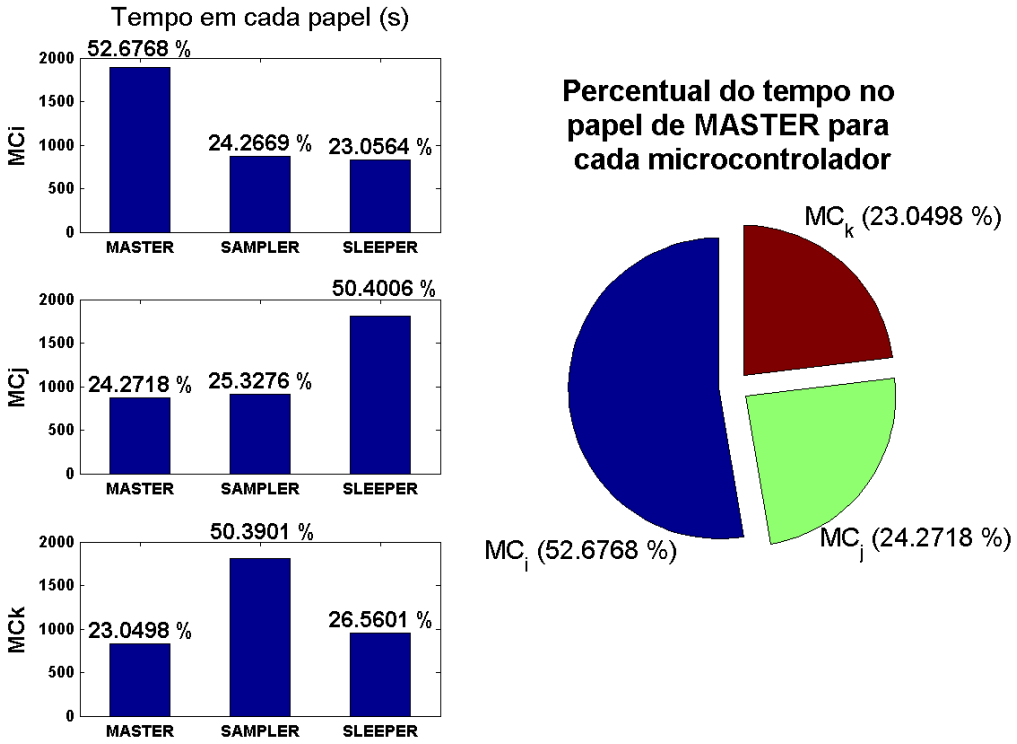


Figura 55 – Distribuição dos papeis de cada microcontrolador para $\lambda = 0,37$ SEU/s.

enviados ao OBC foi de $10,25$ %. A distribuição dos papéis entre os microcontroladores em uma das simulações é mostrada na Figura 56.

Novamente se observa um aumento na distribuição dos papéis entre os microcontroladores, embora o MC_i ainda tenha se mantido no papel de MASTER por mais tempo do que os demais.

Por fim, para uma taxa de falhas 75 vezes maior do que a taxa de referência ($\lambda = 1,11$ SEU/s), a disponibilidade média foi de aproximadamente $99,33$ %, em um intervalo de confiança entre $97,86$ % e 100 % a um nível de confiança de 95 %. Para esse valor λ o percentual médio de valores incorretos enviados ao OBC foi de $16,63$ %. Além de uma redução na disponibilidade e aumento na quantidade de erros nos valores enviados, observa-se um aumento no desvio padrão para um valor superior de λ, o que resultou em um intervalo de confiança alargado.

A distribuição dos papéis entre os microcontroladores em uma das simulações para $\lambda = 1,11$ SEU/s é mostrada na Figura 57.

Para um valor de λ 75 vezes maior a distribuição entre papeis já é praticamente homogênea.

A Tabela 3 resume os resultados obtidos para diferentes valores de λ e apresenta os resultados obtidos por Rosa.

Observa-se que ambas abordagens indicaram disponibilidade de 100 % e taxa de erro

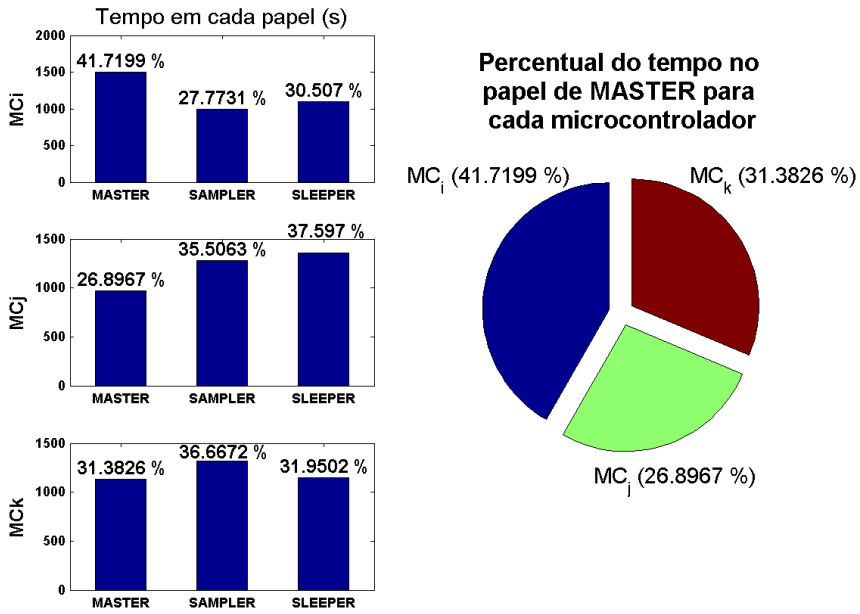


Figura 56 – Distribuição dos papeis de cada microcontrolador para $\lambda = 0,74$ SEU/s.

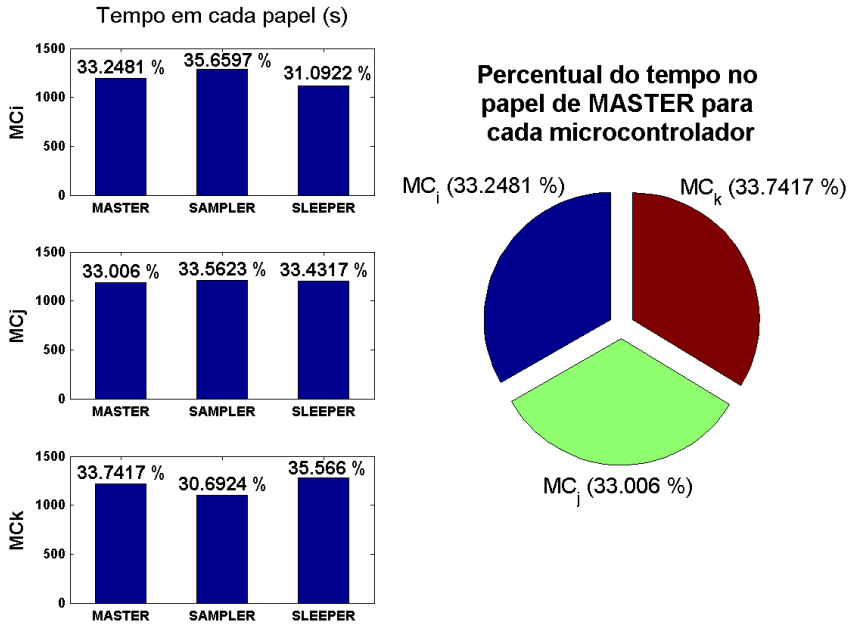


Figura 57 – Distribuição dos papeis de cada microcontrolador para $\lambda = 1,11$ SEU/s.

de 0 % em uma hora de operação do SDATF para a taxa de referência $\lambda = 0,0148$ SEU/s. Para valores superiores da taxa de falhas percebe-se uma leve degradação na disponibilidade e um aumento na taxa de erro observada para os dois casos.

Os valores de disponibilidade para a abordagem baseada em modelagem, embora pró-

Tabela 3 – Resumo dos resultados obtidos na campanha de injeção de falhas

λ (SEU/s)	Disponibilidade na simulação	Intervalo de confiança	Taxa de erro	Disponibilidade observada em (ROSA, 2014)	Taxa de erro observada em (ROSA, 2014)
$0,0148$	100 %	(100 - 100) %	0 %	100 %	0 %
$0,37$	99,94 %	(99,91 - 99,98) %	1,51 %	100 %	1,28 %
$0,74$	99,88 %	(99,85 - 99,91) %	10,25 %	100 %	15,28 %
$1,11$	99,33 %	(97,86 - 100) %	16,63 %	97,62 %	19,65 %

ximos aos obtidos com a metodologia CEU usada por Rosa, se mostram um pouco mais conservadores. No que se refere à taxa de erro nos valores enviados pelo SDATF, as duas abordagens indicaram valores semelhantes.

Capítulo 7

Conclusões e Considerações Finais

7.1 Considerações sobre o trabalho

Ao longo dessa dissertação foi descrito um procedimento completo para a validação de um sistema tolerante a falhas do tipo *Single Event Upset*, baseando-se em modelagem e simulação. O intuito desse trabalho é que tal procedimento possa ser utilizado em diversas soluções, não se restringindo ao Sistema de Determinação de Atitude com Tolerância a Falhas (SDATF), que foi utilizado aqui como uma instância do problema de validação a ser resolvido. Os conceitos aqui descritos, portanto, podem ser utilizados em outros projetos, criando-se um novo modelo de simulação mas baseando-se nas mesmas ideias.

Os experimentos e resultados apresentados no Capítulo 6 reforçam a ideia de que o procedimento proposto pode de fato ser utilizado na validação de uma solução tolerante a falhas - os valores da disponibilidade e da taxa de erros encontrados na campanha de injeção de falhas e simulação foram semelhantes aos encontrados anteriormente numa campanha de injeção de falhas por meio do método *Coding Emulating Upset* (CEU) em um protótipo real[1].

A abordagem por simulação, no entanto, tem suas limitações. Quanto maior a complexidade da solução a ser validada, maior e mais complexo será seu modelo devido ao número elevado de estados, o que aumenta o custo computacional da simulação. Além disso, o processo de injeção de falhas, apesar de não ser intrusivo (visto que não interfere nos tempos de execução da aplicação a ser validada), também agrega custo computacional à simulação. Como consequência, quanto maior o número de falhas injetadas, maior será o tempo gasto para simular a operação do sistema proposto. Sabendo-se que o número de falhas injetadas numa campanha depende de dois fatores - da taxa de falhas λ e do tempo

[1] No trabalho de Rosa não há informação sobre o intervalo de confiança e o número de execuções do experimento para cada valor de λ. Dessa forma, comparou-se a média de disponibilidade e taxa de erros dos experimentos apresentados no Capítulo 6 com os valores apresentados pelo autor em seu trabalho. Apesar de pequenas diferenças nos valores medidos, ambos os procedimentos indicaram disponibilidade total para a taxa nominal de falhas. Além disso, os resultados obtidos na abordagem deste trabalho foram mais conservadores do que os resultados obtidos no protótipo real, o que associa mais confiabilidade ao procedimento aqui proposto.

de operação simulado - conclui-se que a abordagem por simulação é limitada em custo computacional para valores elevados de λ ou para um tempo longo de operação simulado.

Devido a essa limitação em custo computacional, deve-se ressaltar que, no contexto de validação de uma solução tolerante a falhas do tipo SEU, a abordagem aqui apresentada deve ser usada como validação preliminar ou para validação rápida de modificações em soluções.

O tempo de operação usado nesse procedimento, que no experimento realizado para o SDATF foi de uma hora, é pequeno se comparado ao tempo de operação de um satélite, que pode durar alguns meses ou anos. Alternativamente, a abordagem por modelagem e simulação também pode ter seu uso na detecção ou antecipação de problemas de operação do sistema.

No que se refere ao SDATF, a análise dos resultados apresentados na Seção 6.5 permite concluir que a solução é válida, apresentando disponibilidade de 100 % e nenhum erro nos valores de atitude para a taxa de falhas prevista para a baixa órbita. Além disso, para valores muito superiores a essa taxa o SDATF apresenta alta disponibilidade, embora inferior a 100 %.

A contextualização feita no Capítulo 2 é muito importante antes de se fazer a modelagem, pois somente conhecendo o contexto real de ocorrência das falhas é possível os aspectos relevantes na modelagem.

Retomando a Seção 1.2 enunciou-se que o objetivo desse trabalho seria propor um procedimento baseado em modelagem para a validação de uma solução tolerante a falhas do tipo SEU, executando esse procedimento para o SDATF. Diante do que foi discorrido ao longo do trabalho, conclui-se que esse objetivo foi de fato realizado.

7.2 Trabalhos Futuros

O procedimento aqui realizado faz parte do processo de desenvolvimento do SDATF como um protótipo experimental a ser enviado a baixa órbita. O NanosatC-Br2, por se tratar de um satélite científico, consistirá em um novo passo na validação do SDATF no ambiente real.

Além da validação do SDATF pela operação em baixa órbita, outros aspectos deste trabalho podem ser explorados em trabalhos futuros:

- Estudo dos *logs* das simulações com o intuito de analisar os casos mais críticos de ocorrência de falha;
- Busca por sequências de falhas que possam ser críticas à operação do SDATF;
- Uso de métodos formais para provar propriedades da solução, aproveitando a descrição do sistema com base em um formalismo;

- ❏ Comparação da disponibilidade medida na simulação com outros métodos de validação além de *Coding Emulating Upset*;
- ❏ Comparação dos resultados obtidos na simulação com os da metodologia CEU para outros microcontroladores visando comprovar a independência de plataforma.

7.3 Publicação

Parte dos resultados deste trabalho foram submetidos e apresentados no seguinte evento científico:

- ❏ 2^{nd} *IAA Latin American CubeSat Workshop*
  Local: Florianópolis, SC - Brasil
  Datas: 28/02/2016 a 02/03/2016
  Nome da publicação: *Behavior Modelling and Simulation of a Fault Tolerant Attitude Determination System for the NanosatC-Br2*

Referências

ALUR, R. et al. Symbolic analysis for improving simulation coverage of simulink/stateflow models. In: ACM. **Proceedings of the 8th ACM international conference on Embedded software**. [S.l.], 2008. p. 89–98.

ANDJELKOVIC, M. et al. Simulation-based analysis of the single event transient response of a single event latchup protection switch. In: IEEE. **Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2015 IEEE 18th International Symposium on**. [S.l.], 2015. p. 255–258.

AVIŽIENIS, A. et al. Basic concepts and taxonomy of dependable and secure computing. **Dependable and Secure Computing, IEEE Transactions on**, IEEE, v. 1, n. 1, p. 11–33, 2004.

BALASUBRAMANIAN, A. et al. Pulsed laser single-event effects in highly scaled cmos technologies in the presence of dense metal coverage. **Nuclear Science, IEEE Transactions on**, IEEE, v. 55, n. 6, p. 3401–3406, 2008.

BAUMANN, R. et al. Boron as a primary source of radiation in high density drams. In: IEEE. **VLSI Technology, 1995. Digest of Technical Papers. 1995 Symposium on**. [S.l.], 1995. p. 81–82.

BIASI, S. C. de; GATTASS, M. Utilização de quatérnios para representação de rotações em 3d. **Relatório técnico, TecGraf–Pontifícia Universidade Católica do Rio de Janeiro, PUCRIO. Disponível em< http://www. tecgraf. pucrio. br/˜ mgattass**, 2007.

BINDER, D.; SMITH, E.; HOLMAN, A. Satellite anomalies from galactic cosmic rays. **Nuclear Science, IEEE Transactions on**, IEEE, v. 22, n. 6, p. 2675–2680, 1975.

BOUDENOT, J.-C. Radiation space environment. In: **Radiation Effects on Embedded Systems**. [S.l.]: Springer, 2007. p. 1–9.

BOYLESTAD, R. L.; NASHELSKY, L. **Dispositivos eletrônicos e teoria de circuitos**. [S.l.]: Prentice-Hall do Brasil, 1984. v. 6.

CALANDRA, V. P. Applying advanced digital simulation techniques in designing fault tolerant systems. In: IEEE. **Aerospace and Electronics Conference, 1991. NAECON 1991., Proceedings of the IEEE 1991 National**. [S.l.], 1991. p. 508–514.

CHEMINET, A. et al. Characterization of the neutron environment and see investigations at the cern-eu high energy reference field and at the pic du midi. **Nuclear Science, IEEE Transactions on**, IEEE, v. 60, n. 4, p. 2411–2417, 2013.

CHUNBO, W.; YANLIN, Z. The application of the mutual control communication technology in intelligent building safety system. In: IEEE. **E-Product E-Service and E-Entertainment (ICEEE), 2010 International Conference on**. [S.l.], 2010. p. 1–3.

CILLIERS, P.; OPPERMAN, B.; MEYER, R. Investigation of ionospheric scintillation over south africa and the south atlantic anomaly using gps signals: First results. In: IEEE. **Geoscience and Remote Sensing Symposium, 2009 IEEE International, IGARSS 2009**. [S.l.], 2009. v. 2, p. II–879.

COOK, W. R. et al. Pet: A proton/electron telescope for studies of magnetospheric, solar, and galactic particles. **Geoscience and Remote Sensing, IEEE Transactions on**, IEEE, v. 31, n. 3, p. 565–571, 1993.

COTRONEO, D. et al. Experimental analysis of binary-level software fault injection in complex software. In: IEEE. **Dependable Computing Conference (EDCC), 2012 Ninth European**. [S.l.], 2012. p. 162–172.

DIEBEL, J. Representing attitude: Euler angles, unit quaternions, and rotation vectors. **Matrix**, v. 58, p. 15–16, 2006.

DODD, P. E.; MASSENGILL, L. W. Basic mechanisms and modeling of single-event upset in digital microelectronics. **Nuclear Science, IEEE Transactions on**, IEEE, v. 50, n. 3, p. 583–602, 2003.

DUARTE, R. O. et al. An attitude determination system implementation to low orbit small satellite with fault-tolerant techniques. In: IEEE. **On-Line Testing Symposium, 2008. IOLTS'08. 14th IEEE International**. [S.l.], 2008. p. 85–90.

DUARTE R. O.; TORRES, F. E. G. T. H. M.-F. L. S.; KUGA, H. K. A fault-tolerant attitude determination system based on cots devices. In: IAA. **8th IAA Symposium on Small Satellites for Earth Observation**. [S.l.], 2011.

FAURE, F.; VELAZCO, R.; PERONNARD, P. Single-event-upset-like fault injection: a comprehensive framework. **IEEE transactions on nuclear science**, Institute of Electrical and Electronics Engineers, v. 52, n. 6, p. 2205–2209, 2005.

FEILER, P. H.; GLUCH, D. P.; HUDAK, J. J. **The architecture analysis & design language (AADL): An introduction**. [S.l.], 2006.

FERREIRA, A. J. et al. Procedimento experimental para a determinação de atitude de satélites artificiais. 2008.

FRIEDENTHAL, S.; MOORE, A.; STEINER, R. **A practical guide to SysML: the systems modeling language**. [S.l.]: Morgan Kaufmann, 2014.

GARG, V. K. Implementing fault-tolerant services using state machines: Beyond replication. In: **Distributed Computing**. [S.l.]: Springer, 2010. p. 450–464.

GILL, A. et al. Introduction to the theory of finite-state machines. McGraw-Hill, 1962.

GOKA, T. et al. The on-orbit measurements of single event phenomena by eta-v spacecraft. **Nuclear Science, IEEE Transactions on**, IEEE, v. 38, n. 6, p. 1693–1699, 1991.

GRECKI, M.; JABLONSKI, G.; MAKOWSKI, D. Improvements of seu tolerance by spatial redundancy in digital circuits. In: IEEE. **Mixed Design of Integrated Circuits & Systems, 2009. MIXDES'09. MIXDES-16th International Conference**. [S.l.], 2009. p. 123–128.

GUENZER, C.; WOLICKI, E.; ALLAS, R. Single event upset of dynamic rams by neutrons and protons. **Nuclear Science, IEEE Transactions on**, IEEE, v. 26, n. 6, p. 5048–5052, 1979.

HALL, C. D. Spacecraft attitude dynamics and control. **Lecture Notes posted on Handouts page [online]**, v. 12, 2003.

HAREL, D. Statecharts: A visual formalism for complex systems. **Science of computer programming**, Elsevier, v. 8, n. 3, p. 231–274, 1987.

HAREL, D. et al. On the formal semantics of statecharts. 1987.

HAREL D., P. A. On the development of reactive systems. In: SPRINGER-VERLAG. **Logics and Models of Concurrent Systems (K. R. Apt, ed.), NATO ASI Series, Vol. F-13**. [S.l.], 1985. p. 477–498.

HELVAJIAN, H.; JANSON, S. W. **Small satellites: past, present, and future**. [S.l.]: Aerospace Press, 2008.

HOLT, H. M. Assessment of fault-tolerant computing systems at nasa's langley research center. In: IEEE. **Aerospace Conference, 1997. Proceedings., IEEE**. [S.l.], 1997. v. 2, p. 541–549.

HSUEH, M.-C.; TSAI, T. K.; IYER, R. K. Fault injection techniques and tools. **Computer**, IEEE, v. 30, n. 4, p. 75–82, 1997.

JACOBS, A.; WULF, N.; GEORGE, A. D. Task scheduling for reconfigurable systems in dynamic fault-rate environments. In: IEEE. **High Performance Extreme Computing Conference (HPEC), 2013 IEEE**. [S.l.], 2013. p. 1–6.

JR, J. R. S.; MORENO, W.; FALQUEZ, F. Validating fault tolerant designs using laser fault injection (lfi). In: IEEE. **Defect and Fault Tolerance in VLSI Systems, 1997. Proceedings., 1997 IEEE International Symposium on**. [S.l.], 1997. p. 175–183.

KALASHNIKOV, O. A. Statistical variations of integrated circuits radiation hardness. In: **2011 12th European Conference on Radiation and Its Effects on Components and Systems**. [S.l.: s.n.], 2011.

KANAWATI, G. A.; KANAWATI, N. A.; ABRAHAM, J. A. Ferrari: A flexible software-based fault and error injection system. **Computers, IEEE Transactions on**, IEEE, v. 44, n. 2, p. 248–260, 1995.

KEYMEULEN, D. et al. Fault-tolerant evolvable hardware using field-programmable transistor arrays. **Reliability, IEEE Transactions on**, IEEE, v. 49, n. 3, p. 305–316, 2000.

KOLASINSKI, W. et al. Simulation of cosmic-ray induced soft errors and latchup in integrated-circuit computer memories. **Nuclear Science, IEEE Transactions on**, IEEE, v. 26, n. 6, p. 5087–5091, 1979.

KOZAK, T.; MAKOWSKI, D.; NAPIERALSKI, A. Fmc-based neutron and gamma radiation monitoring module for xtca applications. In: IEEE. **Mixed Design of Integrated Circuits and Systems (MIXDES), 2012 Proceedings of the 19th International Conference**. [S.l.], 2012. p. 152–155.

LEE, E. A.; SESHIA, S. A. **Introduction to embedded systems: A cyber-physical systems approach**. [S.l.]: Lee & Seshia, 2011.

LIM, T. S.; MARTIN, R. L.; HUGHES, H. L. Circuit design for nuclear radiation test of cmos multiplier chips. **Circuits and Devices Magazine, IEEE**, IEEE, v. 2, n. 5, p. 29–33, 1986.

LUI, A. T. Tutorial on geomagnetic storms and substorms. **Plasma Science, IEEE Transactions on**, IEEE, v. 28, n. 6, p. 1854–1866, 2000.

LYONS, R. E.; VANDERKULK, W. The use of triple-modular redundancy to improve computer reliability. **IBM Journal of Research and Development**, IBM, v. 6, n. 2, p. 200–209, 1962.

MAIZ, J. et al. Characterization of multi-bit soft error events in advanced srams. In: IEEE. **Electron Devices Meeting, 2003. IEDM'03 Technical Digest. IEEE International**. [S.l.], 2003. p. 21–4.

MARSHALL, P. et al. Autonomous bit error rate testing at multi-gbit/s rates implemented in a 5am sige circuit for radiation effects self test (crest). **Nuclear Science, IEEE Transactions on**, IEEE, v. 52, n. 6, p. 2446–2454, 2005.

MAY, T. C.; WOODS, M. H. Alpha-particle-induced soft errors in dynamic memories. **Electron Devices, IEEE Transactions on**, IEEE, v. 26, n. 1, p. 2–9, 1979.

MEHRESH, R.; UPADHYAYA, S. J.; KWIAT, K. A multi-step simulation approach toward secure fault tolerant system evaluation. In: IEEE. **Reliable Distributed Systems, 2010 29th IEEE Symposium on**. [S.l.], 2010. p. 363–367.

MISHRA, P.; PAPPAS, G.; SOKOLSKY, O. Hierarchical hybrid modeling of embedded systems. Citeseer.

MONTGOMERY, D. C.; RUNGER, G. C. **Applied statistics and probability for engineers**. [S.l.]: John Wiley & Sons, 2010.

MUKHERJEE, S. **Architecture design for soft errors**. [S.l.]: Morgan Kaufmann, 2011.

MURATA, T. Petri nets: Properties, analysis and applications. **Proceedings of the IEEE**, IEEE, v. 77, n. 4, p. 541–580, 1989.

NICOLAIDIS, M. **Soft errors in modern electronic systems**. [S.l.]: Springer Science & Business Media, 2010. v. 41.

NORMAND, E. Single event upset at ground level. **IEEE transactions on Nuclear Science**, Citeseer, v. 43, n. 6, p. 2742–2750, 1996.

PARNAS, D. L. On the use of transition diagrams in the design of a user interface for an interactive computer system. In: ACM. **Proceedings of the 1969 24th national conference**. [S.l.], 1969. p. 379–385.

PARSONS, A. M. The swift gamma-ray burst explorer burst alert telescope (bat). In: IEEE. **Nuclear Science Symposium Conference Record, 2001 IEEE**. [S.l.], 2001. v. 4, p. 2382–2386.

PEREZ-CELIS, J. A. et al. Simulation of fault-tolerant space systems based on cots devices with gpss. IEEE, 2014.

PETERSEN, E. Radiation induced soft fails in space electronics. **Nuclear Science, IEEE Transactions on**, IEEE, v. 30, n. 2, p. 1638–1641, 1983.

PICKEL, J. C.; JR, J. B. Cosmic ray induced in mos memory cells. **Nuclear Science, IEEE Transactions on**, IEEE, v. 25, n. 6, p. 1166–1171, 1978.

ROBINSON, D. G.; NEUTS, M. F. An algorithmic approach to increased reliability through standby redundancy. **Reliability, IEEE Transactions on**, IEEE, v. 38, n. 4, p. 430–435, 1989.

ROSA, M. F. **Validação de um Sistema de Determinação de Atitude com Tolerância a Falhas por meio de uma ferramenta de emulação de Single Event Upsets**. 2014. Monografia (Projeto Final de Curso). Universidade Federal de Minas Gerais, Belo Horizonte.

RUANO, O.; MAESTRO, J.; REYES, P. A simulation platform for the study of soft errors on signal processing circuits through software fault injection. Citeseer, 2007.

SAHBANI, A.; PASCAL, J.-C. Simulation of hybrid systems using stateflow. In: **Proceedings of the 14th European Simulation Multiconference on Simulation and Modelling: Enablers for a Better Quality of Life**. [S.l.: s.n.], 2000. p. 271–275.

SCHANNE, S. et al. The eclairs micro-satellite for multi-wavelength studies of gamma-ray burst prompt emission. **Nuclear Science, IEEE Transactions on**, IEEE, v. 52, n. 6, p. 2778–2785, 2005.

SCHIRMEIER, H.; RADEMACHER, L.; SPINCZYK, O. Smart-hopping: Highly efficient isa-level fault injection on real hardware. In: IEEE. **Test Symposium (ETS), 2014 19th IEEE European**. [S.l.], 2014. p. 1–6.

SCHUCH, N. J.; DURÃO, O. C. The brazilian inpe-ufsm nanosatc-br cubesat program.

SHOKROLAH-SHIRAZI, M.; MIREMADI, S. G. Fpga-based fault injection into synthesizable verilog hdl models. In: IEEE. **Secure System Integration and Reliability Improvement, 2008. SSIRI'08. Second International Conference on**. [S.l.], 2008. p. 143–149.

SHUSTER, M. D. Deterministic three-axis attitude determination. **Journal of Astronautical Sciences**, v. 52, n. 3, p. 405–419, 2004.

SHUSTER, M. D.; OH, S. Three-axis attitude determination from vector observations. **Journal of Guidance, Control, and Dynamics**, v. 4, n. 1, p. 70–77, 1981.

SMITH, E. et al. South atlantic anomaly entry and exit as measured by the x-ray timing explorer. In: NASA. **NASA CONFERENCE PUBLICATION**. [S.l.], 1996. p. 249–256.

STASSINOPOULOS, E.; RAYMOND, J. P. The space radiation environment for electronics. **Proceedings of the IEEE**, IEEE, v. 76, n. 11, p. 1423–1442, 1988.

TORRES, F. E. **Desenvolvimento de um Sistema de Emulação de Single Event Upsets em Dispositivos COTS Baseado na Metodologia Code Emulating Upsets**. Dissertação (Mestrado) — Universidade Federal de Minas Gerais, Belo Horizonte, Fevereiro 2013.

TRIPATHI, R. K.; WILSON, J. W.; YOUNGQUIST, R. C. Electrostatic active radiation shielding-revisited. In: IEEE. **Aerospace Conference, 2006 IEEE**. [S.l.], 2006. p. 9–pp.

VARGAS, V. et al. Evaluating seu fault-injection on parallel applications implemented on multicore processors. In: IEEE. **Circuits & Systems (LASCAS), 2015 IEEE 6th Latin American Symposium on**. [S.l.], 2015. p. 1–4.

VELAZCO, R.; PERONNARD, P.; HUBERT, G. Real-life seu experiments on 90 nm srams in atmospheric environment: measures vs. predictions done by means of musca sep 3 platform. **IEEE Trans. Nucl. Sci**, v. 56, n. 6, p. 3450–3455, 2009.

VELAZCO, R.; REZGUI, S.; ECOFFET, R. Predicting error rate for microprocessor-based digital architectures through ceu (code emulating upsets) injection. **Nuclear Science, IEEE Transactions on**, IEEE, v. 47, n. 6, p. 2405–2411, 2000.

WALLMARK, J.; MARCUS, S. Minimum size and maximum packing density of nonredundant semiconductor devices. **Proceedings of the IRE**, IEEE, v. 50, n. 3, p. 286–298, 1962.

WANG, F.; AGRAWAL, V. D. Single event upset: An embedded tutorial. In: IEEE. **VLSI Design, 2008. VLSID 2008. 21st International Conference on**. [S.l.], 2008. p. 429–434.

ZHANG, Y.; LIU, B.; ZHOU, Q. A dynamic software binary fault injection system for real-time embedded software. In: IEEE. **Reliability, Maintainability and Safety (ICRMS), 2011 9th International Conference on**. [S.l.], 2011. p. 676–680.

ZHOU, Y.; YANG, J.; WANG, Y. Maximizing transient availability of real-time onboard reconfigurable processing platforms: An analytical redundancy inspired approach. In: IEEE. **Information and Automation, 2008. ICIA 2008. International Conference on**. [S.l.], 2008. p. 1246–1251.

ZIEGLER, J. F. et al. Ibm experiments in soft fails in computer electronics (1978–1994). **IBM journal of research and development**, IBM Corp., v. 40, n. 1, p. 3–18, 1996.

ZIEGLER, J. F.; LANFORD, W. Effect of cosmic rays on computer memories. **Science**, American Association for the Advancement of Science, v. 206, n. 4420, p. 776–788, 1979.

APÊNDICE **A**

Cálculo do Intervalo de Confiança nos Experimentos

O procedimento para obtenção do intervalo de confiança para os experimentos deste trabalho será brevemente descrito a seguir. Mais detalhes podem ser obtidos em (MONTGOMERY; RUNGER, 2010).

Seja $x_1, x_2, ..., x_n$ uma amostra de tamanho n de uma variável aleatória X. A estimativa $\bar{x}$ da média desse processo é dada por

$$\bar{x} = \frac{x_1 + x_2 + ... + x_n}{n} = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{17}$$

A estimativa s^2 para a variância desse processo é dada por

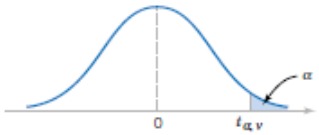
$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2. \tag{18}$$

Seja X definido com distribuição normal. É possível estimar um intervalo de confiança com $100(1-\alpha)$ % de certeza (α é, portanto, o parâmetro utilizado para definir o percentual de certeza ao estimar o intervalo de confiança) em torno da média $\bar{x}$.

O intervalo de confiança pode ser determinado com base na curva de distribuição t, cuja tabela de referência é mostrada na Figura 58. O intervalo de confiança é estimado a partir da Equação 19.

$$\bar{x} - t_{\alpha/2,\nu}s/\sqrt{n} \leq \bar{x} \leq \bar{x} + t_{\alpha/2,\nu}s/\sqrt{n}. \tag{19}$$

Nessa equação, $\nu = n - 1$ é o *grau de liberdade* da amostra. O intervalo de é obtido a partir das estimativas da média ($\bar{x}$) e da variância (s^2) do processo. O termo $t_{\alpha/2,\nu}$ refere-se de uma transformação da curva de distribuição t a partir da tabela de referência da Figura 58, usando os parâmetros α e ν.



Percentage Points $t_{\alpha,\nu}$ of the t Distribution

ν \ α	.40	.25	.10	.05	.025	.01	.005	.0025	.001	.0005
1	.325	1.000	3.078	6.314	12.706	31.821	63.657	127.32	318.31	636.62
2	.289	.816	1.886	2.920	4.303	6.965	9.925	14.089	23.326	31.598
3	.277	.765	1.638	2.353	3.182	4.541	5.841	7.453	10.213	12.924
4	.271	.741	1.533	2.132	2.776	3.747	4.604	5.598	7.173	8.610
5	.267	.727	1.476	2.015	2.571	3.365	4.032	4.773	5.893	6.869
6	.265	.718	1.440	1.943	2.447	3.143	3.707	4.317	5.208	5.959
7	.263	.711	1.415	1.895	2.365	2.998	3.499	4.029	4.785	5.408
8	.262	.706	1.397	1.860	2.306	2.896	3.355	3.833	4.501	5.041
9	.261	.703	1.383	1.833	2.262	2.821	3.250	3.690	4.297	4.781
10	.260	.700	1.372	1.812	2.228	2.764	3.169	3.581	4.144	4.587
11	.260	.697	1.363	1.796	2.201	2.718	3.106	3.497	4.025	4.437
12	.259	.695	1.356	1.782	2.179	2.681	3.055	3.428	3.930	4.318
13	.259	.694	1.350	1.771	2.160	2.650	3.012	3.372	3.852	4.221
14	.258	.692	1.345	1.761	2.145	2.624	2.977	3.326	3.787	4.140
15	.258	.691	1.341	1.753	2.131	2.602	2.947	3.286	3.733	4.073
16	.258	.690	1.337	1.746	2.120	2.583	2.921	3.252	3.686	4.015
17	.257	.689	1.333	1.740	2.110	2.567	2.898	3.222	3.646	3.965
18	.257	.688	1.330	1.734	2.101	2.552	2.878	3.197	3.610	3.922
19	.257	.688	1.328	1.729	2.093	2.539	2.861	3.174	3.579	3.883
20	.257	.687	1.325	1.725	2.086	2.528	2.845	3.153	3.552	3.850
21	.257	.686	1.323	1.721	2.080	2.518	2.831	3.135	3.527	3.819
22	.256	.686	1.321	1.717	2.074	2.508	2.819	3.119	3.505	3.792
23	.256	.685	1.319	1.714	2.069	2.500	2.807	3.104	3.485	3.767
24	.256	.685	1.318	1.711	2.064	2.492	2.797	3.091	3.467	3.745
25	.256	.684	1.316	1.708	2.060	2.485	2.787	3.078	3.450	3.725
26	.256	.684	1.315	1.706	2.056	2.479	2.779	3.067	3.435	3.707
27	.256	.684	1.314	1.703	2.052	2.473	2.771	3.057	3.421	3.690
28	.256	.683	1.313	1.701	2.048	2.467	2.763	3.047	3.408	3.674
29	.256	.683	1.311	1.699	2.045	2.462	2.756	3.038	3.396	3.659
30	.256	.683	1.310	1.697	2.042	2.457	2.750	3.030	3.385	3.646
40	.255	.681	1.303	1.684	2.021	2.423	2.704	2.971	3.307	3.551
60	.254	.679	1.296	1.671	2.000	2.390	2.660	2.915	3.232	3.460
120	.254	.677	1.289	1.658	1.980	2.358	2.617	2.860	3.160	3.373
∞	.253	.674	1.282	1.645	1.960	2.326	2.576	2.807	3.090	3.291

ν = degrees of freedom.

Figura 58 – Tabela de referência da distribuição t.

Fonte: (MONTGOMERY; RUNGER, 2010)

EXEMPLO

Seja, por exemplo, uma variável aleatória X, definida a partir de um processo estocástico com distribuição normal de probabilidade.

Suponha que um experimento sobre esse processo levante a seguinte amostra: $x_1 = 6,63; x_2 = 6,77; x_3 = 6,72; x_4 = 6,88; x_5 = 6,51; x_6 = 6,57$. Considerando o tamanho da amostra $n = 6$, a estimativa da média é dada por

$$\bar{x} = \frac{6,63 + 6,77 + 6,72 + 6,88 + 6,51 + 6,57}{6} = 6,68. \tag{20}$$

A estimativa para a variância é dada por

$$s^2 = \frac{(6{,}63-6{,}68)^2+(6{,}77-6{,}68)^2+(6{,}72-6{,}68)^2+(6{,}88-6{,}68)^2+(6{,}51-6{,}68)^2+(6{,}57-6{,}68)^2}{6-1} = 0,01864. \quad (21)$$

Suponha que se deseje definir o intervalo de confiança com 95% de certeza. Nesse caso temos $100(1-\alpha) = 95 \rightarrow \alpha = 0,05$, e $\nu = n-1 = 5$. Consultando a tabela da Figura 58, temos

$$t_{\alpha/2,\nu} = t_{0,025,5} = 2,571. \quad (22)$$

Substituindo os valores na Equação 19, encontra-se o intervalo de confiança:

$$6,536699 \leq \bar{x} \leq 6,823301. \quad (23)$$

Isso significa que o valor médio da variável aleatória X deverá estar entre $6,536699$ e $6,823301$ com 95% de certeza.

Anexos

ANEXO A

CBCnews | Technology & Science

Home | World | Canada | Politics | Business | Health | Arts & Entertainment | Technology & Science | Trending | Weather | Video

Technology & Science | Quirks & Quarks Blog | Spark | Photo Galleries

Solar storm knocks out flight control systems in Sweden, grounds planes

Flights disappeared from radar screens

The Associated Press Posted: Nov 04, 2015 3:38 PM ET | Last Updated: Nov 04, 2015 3:40 PM ET

Aviation officials say a solar storm knocked out the air traffic control systems in Sweden on Wednesday, prompting them to close the country's airspace for more than an hour.

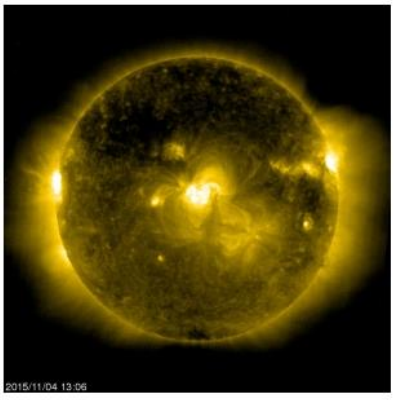
The civil aviation authority said the solar storm created disturbances in the Earth's magnetic field, which affected radar installations in southern Sweden. No such problems were reported in neighbouring countries.

Agency spokesman Per Froberg said flights disappeared from radar screens in Swedish air traffic control towers during the blackout, which lasted about an hour until 5:30 p.m. local time (11:30 a.m. ET). Froberg said it was unclear why the impact was so severe, adding the last time something similar happened in Sweden was in 1999.

"We're working on sorting out the delays. We can't examine the cause right now. We have our hands full," he said.

He couldn't say how many flights were affected, but the country's main airports listed dozens of delays.

Parked aircraft are seen at at Stockholm Arlanda Airport in a photo from March 2015. Swedish airspace was closed for an hour Wednesday after a solar storm knocked out air traffic control systems. (Johan Nilsson/Associated Press)



An image from NASA and ESA's Solar and Heliospheric Observatory shows the sun as it appeared today. An unusually fast stream of solar wind with a series of shock waves has been hitting the Earth, creating auroras and polar geomagnetic storms. (NASA)

Air traffic control officials in neighbouring Denmark and Finland say they didn't experience any problems.

"There haven't been any disturbances. Only a few delays in Copenhagen because of the problems in Stockholm," said Mette Just of Naviair, Denmark's air navigation service.